The Dissertation Committee for Narameth Nananukul Certifies that this is the approved version of the following dissertation:

Lot-Sizing and Inventory Routing for a Production-Distribution Supply Chain

Committee:

Jonathan F. Bard, Supervisor

J. Wesley Barnes

David P. Morton

Erhan Kutanoglu

Leon Lasdon

Lot-Sizing and Inventory Routing
for
a Production-Distribution Supply Chain

by

Narameth Nananukul, B. Eng.; M. Eng.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2008

Dedication

This dissertation is dedicated to my father.

Acknowledgements

I like to thank several people for supporting my study at the University of Texas at Austin. First and foremost I like to thank Dr. Jonathan F. Bard, who serves as my advisor. With his expertise in the field, he has given invaluable advice for my research.

My thanks also go to all the committee members, Dr. Wesley Barnes, Dr. David Morton, Dr. Erhan Kutanoglu, and Dr. Leon Lasdon, for their invaluable comments and suggestion.

Last, but not least, I would like to thank my wife Aei for her support and love during the past few years. I am also grateful to my parents for their love and support.

Lot-Sizing and Inventory Routing
for
a Production-Distribution Supply Chain

Narameth Nananukul, PhD

The University of Texas at Austin, 2008

Supervisor: Jonathan F. Bard

The integration of production and distribution decisions presents a challenging problem for manufacturers trying to optimize their supply chain. At the planning level, the immediate goal is to coordinate production, inventory, and delivery to meet customer demand so that the corresponding costs are minimized. Achieving this goal provides the foundations for streamlining the logistics network and for integrating other operational and financial components of the system. In this paper, a model is presented that includes a single production facility, a set of customers with time varying demand, a finite planning horizon, and a fleet of vehicles for making the deliveries. Demand can be satisfied from either inventory held at the customer sites or from daily product distribution.

A procedure centering on a reactive tabu search is developed for solving the full problem. After a solution is found, path relinking is applied to improve the results. A novel feature of the methodology is the use of an allocation model in the form of a mixed integer program to find good feasible solutions that serve as starting points for the tabu search. Lower bounds on the optimum are obtained by solving a modified version of the allocation model. Computational testing on a set of 90 benchmark instances with up to 200 customers and 20 time periods demonstrates the effectiveness of the approach. In all cases, improvements ranging from 10 – 20% were realized when compared to those obtained from an existing greedy randomized adaptive search procedure (GRASP). This often came at a three- to five-fold increase in runtime, however.

A hybrid scheme that combines the features of reactive tabu search algorithm and branch-and-price algorithm is also developed. The combined approach takes advantage of the efficiency of the tabu search heuristic and the precision of the branch-and-price algorithm. Branching strategy that is suitable for the problem is proposed. Several advance techniques such as column generation heuristic and rounding heuristic are also implemented to improve the efficiency of the algorithm. Computational testing on standard data sets shows that a hybrid algorithm can practically solve instances with up to 50 customers and 8 time periods which is not possible by standard branch-and-price algorithm alone.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 MOTIVATION

The role that the customer plays in logistics management is becoming increasingly important as the areas in which businesses compete expand to include quality, on time delivery, warranty and repair services, pricing contracts, and remanufacturing. Many companies now realize that greater value can be offered to their customers by effectively integrating logistics management and product availability to improve the timeliness and consistency of delivery. Achieving a level of integration that will yield new benefits requires that the production and distribution decisions be made daily to balance setup, holding and delivery costs in an optimal framework. These decisions have traditionally been made separately; however, their integration can have a significant impact on the overall system. Chandra and Fisher (1994) have shown, for example, that solving the production scheduling and vehicle routing problems simultaneously, can result in total operating cost reductions from 3 to 20%.

A vendor managed inventory replenishment (VMI) system is a good example of the type of integration mentioned above (Cetinkaya and Lee 2006 and Zhao et al. 2007). In the VMI model a vendor observes and controls the inventory levels of its customers, as opposed to conventional approaches where customers monitor their own inventory and decide the time and amount of products to reorder. One of the benefits of VMI is that vendors can obtain a more uniform utilization of transportation resources, which leads to lower distribution costs. It also offers them the flexibility to choose the most preferred transportation mode. Customers benefit from higher service levels and greater product availability due to the fact that vendors can use existing inventory data at their customer sites to predict future demand more accurately.

In general, the problem of optimally coordinating production and transportation is called the production-inventory-distribution routing problem (PIDRP) (Lei et al. 2006). Addressing these components in a single framework offers a holistic view of the logistics

network and provides a good starting point for the full integration of the supply chain. The PIDRP usually arises in the retail industry where customers or outlets rely on a central supplier or manufacturer to provide them with a given commodity on a regular basis. In the version of the problem addressed in this dissertation, a manufacturer must develop minimum cost production and distribution schedules for a single product that are sufficient to meet all customer demand over the planning horizon. To ensure timely distribution and to avoid shortages, excess production can be stored at either the plant or at the customer sites within some limits; however, inventory cannot be transferred between sites and stockouts are not permitted. It is further assumed that demand is known for each day of the planning horizon and that all initial inventories are zero. In the model, two lot-sizing decisions must be made. The first concerns the production-inventory tradeoff; the second relates to the daily distribution decisions over the planning horizon.

With regard to the latter, deliveries are made routinely by a fleet of homogeneous vehicles that begin and end their run at the plant. In the most complex scenario investigated, a vehicle routing problem (VRP) must be solved daily to determine, in conjunction with the production decisions, which customers to visit and how much product to deliver to each. Due to either vehicle capacity limits or favorable cost tradeoffs, it may be desirable to visit a customer on a day in which ample stocks exist at his site to build up inventory. In fact, this might be the only feasible option if all demand is to be met.

In less restrictive versions of the problem, the routing component is replaced with a clustering and distribution component. Two options are considered. In the first option, customers are clustered or grouped together in a preprocessing step; in the second, the clustering and production lot-sizing decisions are made simultaneously. In either case, a single vehicle is assigned to each cluster and it is assumed that customers are visited in a fixed order. That is, the routing aspect of the problem is dropped, leaving only vehicle capacity as the limiting factor. This is common in outbound logistics network design (Baudin 2004).

To summarize, the objective of the problem investigated in this research is to minimize a combination of holding and distribution costs over the planning horizon without incurring any stockouts at the customer sites. In so doing, three critical decisions have to be made:

- how many items to manufacture each day
- when to visit each customer
- how much to deliver to a customer during each visit
- which delivery routes to use

The PIDRP is different from traditional VRPs because it requires multiple customer visits to satisfy demand spread out over an extended period of time. It is most similar to the inventory routing problem, IRP (Golden et al. 1984, Federgruen and Zipkin 1984, and Dror and Ball 1987) and the periodic routing problem, PRP (Gaudioso and Paletta 1992, Mourgaya and Vanderbeck 2007, and Parthanadee and Logendran 2006). Although there has been much research on these two problems, little of it carries over to the PIDRP, which has only been studied intermittently. The primary reason relates to the formidable complexity of its structure, which is defined by a combination of a capacitated lot-sizing problem and a capacitated, multiperiod VRP. The full problem has so far proven to be beyond the capability of exact methods. By decoupling of the lot-sizing and routing decisions, though, several researcher have had some success in finding good solutions with heuristics. Chandra and Fisher (1994), for example, first determine a production schedule without regard to the logistics. Next, they develop a distribution schedule for each planning period based on the results obtained from the first-stage model. This approach worked well when there was enough inventory in the system to buffer the production from the distribution operations but led to increased holding costs.

The focus of this research is on the design, development, and implementation of several algorithms for solving the PIDRP that give verifiably high quality solutions within acceptable runtimes. For planning purposes, this means within one or two hours. First, a tabu search metaheuristic is developed for a single production facility that manufactures one product that is distributed by a given fleet of identical vehicles. Setup and holding cost are considered at the plant and holding costs at the customer sites. The

tabu search algorithm is distinguished by its neighborhood structure, short- and long-term memory functions, and relinking component.

Next, several exact solution methods are investigated in any effort to improve on the heuristic results. Initially, a column generation algorithm is developed to handle the routing and delivery aspects of the problem. Decomposition of the original problem results in a master problem that does not include routing constraints and a set of subproblems in the form of VRPs, one for each time period. As part of the solution methodology, the metaheuristic is run periodically to provide upper bounds at various nodes in the search tree. These upper bounds help fathom nodes during the implicit enumeration of production and delivery schedules. Advance techniques such as column generation heuristic are also investigated. A wide range of problem instances are generated to test the algorithms.

1.2 GUIDE TO THIS DISSERTATION

The contents of this dissertation are arranged as follows: Chapter 2 extensively reviews the literature related to lot-sizing, vehicle routing and production and inventory problems. In Chapter 3 the model for the PIDRP is introduced, follows by Chapter 4 where tabu search algorithm for the PIDRP is presented. Chapter 5 introduces several adjusted models for lower bound computation. Chapter 6 reports computational results for tabu search algorithm. Chapters 7 and 8 present the branch-and-price algorithm and discuss several issues that cause difficulty for the standard branch-and-price algorithm. Chapters 9 and 10 summarize a number of enhanced features and report computational results for the modified branch-and-price algorithm. Finally, Chapter 11 concludes the dissertation and suggests future work.

Chapter 2

Literature Review

The research on the PIDRP involves three different areas: lot-sizing, vehicle routing, and production and inventory. Over the last four decades, an enormous amount has been written in each of these areas. In this section, we briefly review the major developments, focusing on the work that is most relevant to the dissertation.

2.1 LOT SIZING

The uncapacitated single-item lot-sizing model was first introduced by Wagner and Whitin (1959) who developed the now classic dynamic programming (DP) algorithm to find solutions. Soon after, the basic model was extended in many directions to include backlogging, holding and production capacities, start-up constraints, and production in series. A formal description of these variants is given by Wolsey (1998).

Network representations of the uncapacitated lot-sizing problem provide an easy way to visualize its components and to begin designing algorithms. For small instances, the optimal policy can be found by enumerating all paths through the network and choosing the one with minimum cost (Wolsey 1998). For more complex situations, solving a shortest path problem will not provide the solution; more sophisticated algorithms are required. In particular, Pochet and Wolsey (1991) developed a MIP-based method for the multi-item lot-sizing problem. They proposed a strong cutting planes technique for solving the problem; Federgruen et al. (1991) proposed a forward algorithm, where the recursion of dynamic programming works from the first period toward the last period, for the general dynamic lot-sizing model; and Gutierrez et al. (2002) solved the dynamic lot-sizing problem with bounded inventory by using a new DP algorithm that is based on newly proposed properties that can be used to characterize optimal plans. The work on lot sizing most relevant to this dissertation is summarized below.

Pochet (1988) described a family of valid inequalities for the single-item capacitated lot-sizing model in its most general form. He also considered a special case with equal production capacity in each period. He showed that under some special conditions the proposed valid inequalities define facets of the polyhedron associated with the relaxed feasible region. The conditions are based on the sum of demands between a pair of periods and the production capacity. The computational results showed that 69 out of 80 problem instances ranging from 12 to 24 periods could be solved to optimality at the root node of the branch and bound tree.

In an expanded version of earlier work, Pochet and Wolsey (1991) considered two classes of multi-item lot-sizing problems. The first class consisted of single stage problems involving joint machine capacity constraints and/or startup costs, and the second consisted of multistage problems with general product structure. In the paper, they presented formulations for the multi-item single stage problem and the multi-item multistage problem as well as a family of strong valid inequalities and separation routines. The inequalities typically defined facets for both problems, but even when this was not the case, testing showed that they provided tight relaxations. To find solutions, they developed a branch-and-cut algorithm in which the valid inequalities were generated with a mathematical programming system called MPSARX.

Subsequently, Pochet and Wolsey (1993) proposed a new algorithm for a version of the classical lot-sizing problem with constant production capacities (LCC) and a variant in which the capacity in each period is an integer multiple of some basic batch size (LCB). They showed that the classical dynamic program for LCC simplifies for LCB giving an $O(\tau^2 \min\{\tau, C\})$ algorithm, where n is the number of periods, C the batch size, and τ the number of periods into which the planning horizon is divided. Based on the new algorithm they showed how to formulate both problems as linear programs with $O(\tau^3)$ variables and constraints. Next, they described a class of inequalities for LCB that captured both the dynamic nature of the problem as well as the capacity aspects. Further analysis confirmed that these inequalities are inclusive of all the known classes of valid inequalities for LCB.

Federgruen and Tzur (1991) proposed a forward algorithm that solves the general dynamic lot-sizing model in $O(\tau \log \tau)$ time and $O(\tau)$ space, where τ denotes the number of periods into which the planning horizon is divided. Linear, i.e., $O(\tau)$-time and space algorithms were developed for two special cases. The first case considers models without speculative motives for carrying stock; namely, instances in which the per unit order cost increases in each interval of time by less than the cost of carrying a unit in stock. In the second case, instances with nondecreasing setup costs were considered. To find solutions, the authors developed a procedure that is based on a well known result for the general lot-sizing problem which states that the optimal order strategy could be determined by a zero-inventory ordering strategy. A zero-inventory ordering strategy is completely determined by the specification of the last order period (the last period in which the product was ordered) preceding any period t. Their procedure is based on the observation that for any pair of periods $k < l$ the difference function $F(k,t) - F(l,t)$ is monotone (in $t \geq l$), where $F(l,t)$ is the minimum cost in the first t periods when the final period in which the product is ordered is period l. So the solution of the model reduces to determining a sequence $\{l(t); t = 1,\ldots,\tau\}$ where τ is the number of periods and $l(t)$ is an optimal period to perform the last ordering when minimizing cost in the first t periods. Their proposed algorithm is a forward algorithm with sequential determination of a pair $(l(j), F(j))$ for $j = 1,\ldots,\tau$ where $F(j)$ represents the minimum cost in the first j periods. Computational experiments showed that the algorithms produced major time savings compared to the results reported by Evans (1985). Large problem instances with 5000 time periods require no more than 1.5 CPU seconds when executed on an IBM 4381. Also, the algorithm outperformed efficient implementations of the Wagner-Whitin method for problem instances with more than 20 periods.

Gutiérrez et al. (2002) investigated a deterministic single-item dynamic lot-sizing problem with time-varying storage capacities and shortages. Concave reorder and holding costs and nonnegative inventory levels were assumed. They derived properties that must be satisfied by at least one optimal plan over a planning horizon of τ periods and used them to construct an $O(\tau^3)$ DP algorithm. The optimal value function $f(t, I_{t-1})$ in

the recursion represents the minimum cost of satisfying demand during periods t through τ if the inventory is I_{t-1} at the beginning of period t, where the pair (t, I_{t-1}) constitutes a state. The number of possible decisions to be considered in the DP is reduced substantially when the decision state set follows the optimal plan properties. The performance of the algorithm was studied using both worst-case and average-case analysis on problem instances with 25 to 500 periods. Comparisons with the more traditional approach (e.g., see Love 1973) showed that the proposed algorithm was an average of 30 times faster. Extensions that allow for shortages and backlogging are discussed by Gutiérrez et al. (2007).

2.2 VEHICLE ROUTING PROBLEMS

The VRP is one of the most studied problems in combinatorial optimization. Since it was first defined by Dantzig and Ramser (1959), innumerable papers have been published describing models, algorithms and variations. The inventory routing problem (IRP) and the periodic vehicle routing problem (PRP) are two important variations that relate to the proposed research. Both of these problems are relaxations of the PIDRP, but differing in several ways. Neither, for example, takes the production decision and inventory level at the plant into consideration. In addition, the PRP assumes that the delivery patterns defined by delivery frequencies or delivery days are given in advance and tries to select the most suitable pattern for each customer. Some of the more popular heuristics and exact methods for routing-type problems are now summarized.

Golden et al. (1984) were the first to investigate the interrelated problem of inventory allocation and vehicle routing. The particular application involved an energy-products company that distributed liquid propane to its customers. They developed a heuristic for designing an integrated delivery planning system. Their objective was to compare the distribution rule used by the company to the proposed heuristic. Approximately 3000 customers were considered in the study. They used the company's historical data to calculate an average consumption rate for each customer and the latest replenishment data to calculate when each customer could be expected to need a re-supply. The next replenishment was scheduled based on this information. The proposed

heuristic included a customer selection algorithm that decides which customers to visit on each day in a cost-effective way. The results were compared to those from a simulation experiment and showed an improvement of 8.4% in the number of gallons/hour, a 50% reduction of stockouts, and a 23% reduction in total costs.

Federgruen and Zipkin (1984) investigated the inventory routing problem by viewing it as a special case of the VRP. They considered a single period model with stochastic demand and nonlinear inventory costs, and developed a nonlinear integer programming model. For any assignment of customers to routes, the model decomposes into an inventory allocation problem and a number of traveling salesman problems (TSPs). Starting with an initial inventory allocation, they iteratively applied interchange procedures to construct a better set of tours and an optimization procedure for improving the inventory allocation. The algorithm terminated when there was no reduction in the combined inventory and routing costs could be found. Extensive testing on problem instances with 50 – 70 customers defined for a single period showed that about 6 – 7% savings in operating costs could be achieved when compared to conventional approaches.

Dror and Ball (1987) developed a model for the inventory allocation and vehicle routing problems over an extended planning horizon. The objective was to minimize the long-term average delivery costs subject to no customers running out of stock at any time. Because the full problem was too large to solve directly, a decomposition heuristic was proposed that had a look-ahead feature to account for costs beyond the current day. This allowed the long-term effects to be transposed to the short-term, thus yielding a much smaller problem to solve. One component of the heuristic included a route improvement scheme developed by Dror et al. (1985) that involved node interchange operations for a given set of routes.

They found solutions to the annual IRP by solving a sequence of two-week problems within a rolling horizon framework. At any point in time the customers were partitioned into two groups, the first consisting of those who needed replenishment in the upcoming week and the second, of those who did not. A feasible schedule was then obtained for the customers in the first group that took into account the vehicle capacity constraints. Next, the route improvement scheme was applied to achieve local

optimality. Given a feasible solution, the neighborhood search proceeded by interchanging the positions of a pair of customers in a route and between different routes with the objective of reducing the total distribution cost over the planning period. Results from the proposed algorithm were compared with those obtained from a simulation study conducted by Dror et al. (1985) and those realized from applying the manual rules in use at the time. The proposed algorithm provided more than a 50% increase in performance over the manual procedures and more than 25% increase over the simulation, where performance measurement was based on the method proposed by Assad et al. (1983).

Chien et al. (1989) worked with a single-day model for the IRP but tried to find less myopic solution by passing inventory information from one day to the next. For a profit maximization objective, a Lagrangian-based procedure was used to generate upper bounds while lower bounds (feasible solutions) were obtained from a two-phase heuristic that used the relaxed solution as input. For a given set of Lagrangian multipliers, the original problem was decomposed into two subproblems: an inventory allocation problem and a customer assignment/vehicle utilization subproblem. Both were solved optimally and a standard subgradient algorithm was employed to identify good multipliers. In phase one of the feasibility heuristic, initial routes are constructed from the inventory allocation and the customer assignments obtained from the subproblem solutions. In phase two, a procedure that ensures feasibility and a routine that checks for possible improvement are applied. Testing on problem instances that contained 20, 25 or 30 customers and a single period indicated that solutions within 1 to 3% of optimality were obtained within 482.57 seconds in all cases.

Gaudioso and Paletta (1992) proposed several algorithms for the PRP when the objective is to minimize the maximum number of vehicles used simultaneously during any day of planning period. In the model, each customer i required the delivery of a specified amount of commodity exactly once in each time interval of T_i days during a planning horizon of τ days. They also required that two successive deliveries be spaced between K_i and U_i days apart. When two or more routes were disjoint, it was permissible to service them with the same vehicle.

The overall approach uses the idea of making allocations one customer at a time. First, the customers are placed on a list in nondecreasing order of their T_i values. Next, beginning with the first customer on the list, each is assigned to the first feasible route, as judged by the capacity and travel time constraints. If the current delivery schedule for the i^{th} customer does not result in an increase in the fleet size, then the assignment is confirmed and the $i+1^{st}$ customer is considered. Otherwise consider the next smallest delivery combination of a customer and an assigned route (in lexicographical order) from among those not comprising days for which an increase in the fleet size has been proved necessary.

Three different variants of the basic algorithm were considered. The first (*BA*1) included a 2-opt procedure for reducing the length of all routes plus a bin-packing heuristic with FFD (first fit decreasing) logic. The second (*BA*2) combined a procedure for exchanging two customers between routes that were active during the same day with a FFD bin-packing procedure. The third algorithm (*BA*3) tried to smooth the assignments by moving selected customers on peak days to adjacent days. Testing was done on a VAX 8200 and with problem instances that contained between 100 and 300 customers. The computational results showed that the basic algorithm provided good starting solutions, but there was significant room for improvement. In general, *BA*3 allowed a partial rescheduling of the customers, while *BA*1 and *BA*2 were only able to improve the existing routes. Also, *BA*3 appeared to uniformly outperform the others with respect to the 24 cases studied, providing a better solution than the basic algorithm in 22 cases, *BA*1 in 14 cases, and *BA*2 in 7 cases.

Gendreau et al. (1994) developed a metaheuristic called *Taburoute* for the VRP with capacity and route length restrictions. The neighborhood structure was defined by all solutions that could be reached from the current solution by removing a vertex from its current route and inserting it into another route containing one of its p nearest neighbors. GENI, a generalized insertion procedure developed by Gendreau et al. (1992) for the TSP, was used for this purpose. During the search process, infeasible solutions with respect to the two major constraints were considered to reduce the likelihood of being trapped at a local minimum. Computational testing on problem instances ranging

from 50 to 199 vertices indicated that the proposed algorithm outperformed the best existing heuristics at that time.

Cordeau et al. (1997) proposed a tabu search heuristic for following three routing problems: the PRP, the periodic TSP, and the multi-depot VRP. A common component of their algorithms was GENI, which was used to perform insertions of unrouted customers, and the removal of customers from their current route and their reinsertions into different routes. In the first instance, GENI gradually constructs a Hamiltonian tour by inserting at each step a vertex v between two of its p closest neighbors. At this point, the neighbors, denoted by v_i and v_j, are not necessarily consecutive, but after insertion the sequence (v_i, v, v_j) will be part of the tour. To perform the insertion, GENI considers 4-opt modifications of the tour around v_i and v_j, and selects the least-cost option. The reverse of this procedure is applied to remove a vertex v from a tour. Computational testing on instances ranging from 48 to 288 customers and from 4 to 6 periods, showed that the proposed heuristic outperformed its competitors on all three problem types.

Kontoravdis and Bard (1997) developed a greedy randomized adaptive search procedure (GRASP) for the vehicle routing problem with time windows (VRPTW). Their primary objective was to find the minimum number of vehicles required to visit a set of customers subject to time window constraints; a secondary objective was to minimize the total distance traveled. In the model they assumed that each customer required a predetermined amount of service in the form of pickups and deliveries and that split service was not permitted. Also, the fleet was assumed to be homogeneous and vehicle had a finite capacity. Their algorithm comprised two phases. In phase one, a set of seed customers is chosen to form initial routes. The remaining customers are iteratively assigned in series according to an adaptive set of rules. During construction, the best feasible insertion location in each route for every unassigned customer is determined. A penalty cost that measures the cost that would have to be paid later if the corresponding customer is not assigned to its current best position is also calculated. Insertions are made based on these two pieces of information. When no feasible assignment exists for a customer, a new route is started.

In phase two, a neighborhood search is performed to find a local optimal. Due to the high computational requirements, this search is applied to the best solution found after every five iterations of phase one rather than to each feasible solution. Also, in phase two, each route is considered for elimination by attempting to move assigned customers from one route to another, with routes having fewer customers examined first. As part of the solution methodology, they also developed three different lower bounding procedures that could be used to gauge the quality of the solutions. The first considered the bin-packing aspect of the problem with regard to vehicle capacity; the second was based on the maximum clique associated with the customers' incompatibility graph; and the third independently exploited the time window constraints.

The full algorithm was tested on benchmark data sets with up to 100 customers (Solomon 1987) and four large industry-based problems in which the number of customers ranged between 100 and 417. The computational results showed that the proposed procedure outperformed current heuristics and that runtimes on the smaller instances were substantially less than the time taken by exact methods. By using the three lower bounding procedures, they were able to confirm that optimal solutions were found in almost all cases involving balanced service and tight capacity constraints and a significant proportion of the remainder.

With respect to exact methods, Desrochers et al. (1992) were the first to develop a column generation-based procedure for the VRPTW. In the model, they assumed a homogeneous fleet and one type of service. The objective was to minimize the distance traveled without violating the time window or vehicle capacity constraints. Rather than trying to minimize the number of vehicles used as well, they placed an upper limit on the fleet size.

The master problem was initially set up as a set partitioning problem but was ultimately replaced with a set covering relaxation to facilitate the computations. The subproblems took the form of elementary shortest path problems with time window and capacity constraints. Difficulty in solving the subproblems led to the development of two DP algorithms which allowed for routes with cycles, but whose worst-case complexity is pseudopolynomial. Infeasibilities were addressed during branch and bound.

Branching was done in two stages. In the first stage, integrality of the number of vehicles and the total distance traveled provided the basis constructing the search tree. In the second stage, branching was performed on the arcs of the subproblem network. Arcs associated with columns in the master problem that contained cycles where chosen first. For each column, a score was given to the arcs incident to nodes visited more than once. The arc with the highest score was chosen and two branches were created, 0 and 1, respectively. If no columns existed that contained cycles, fractional variables were then examined. Testing was performed on Solomon's 29 data sets, each containing 25, 50, and 100 customer instances. The results showed that the column generation algorithm was able to optimally solve all 29 instances involving 25 customers, 14 instances with 50 customers, and 7 instances with 100 customers.

Taking a different approach, Bard et al. (2002) developed a branch-and-cut procedure for the VRPTW. They focused on the problem of finding the minimum number of vehicles required to visit a set of customers subject to time window and capacity constraints. The fleet was assumed to be homogeneous and located at a common depot. First, they presented the mixed-integer linear programming model for the problem and then described the principal classes of valid inequalities that can be used to reduce the solution space of the original formulation after relaxing the integrality constraints. In particular, they made use of capacity inequalities, comb inequalities, box inequalities, and path-box inequalities all of which were originally developed for the TSP and VRP. VRPTW-specific inequalities included subtour elimination constraints, incompatible pair inequalities, and incompatible path inequalities. The branch-and-cut procedure employed a combination of cutting planes, separation routines, and implicit enumeration to find feasible solutions. Testing was done using Solomon's 50 and 100 customer data sets. The results showed that problems instances with 50 customers could routinely be solved to optimality, while the larger instances could also be solved if their time windows were sufficiently tight.

Gaur and Fisher (2004) developed and implemented a system for a PRP at Albert Heijn, a supermarket chain in the Netherlands. In the model, they assumed that there is a single homogeneous product with deterministic but time-varying demand. Although the

application involved the replenishment of several items, all items were replenished from a single distribution center so they were able to formulate an equivalent single-product problem. This was done by expressing the demand rates of all items in units of weight and volume, translating the result into fractions of containers, and aggregating. Each vehicle in the fleet was identical with a capacity of 26 containers.

To find solutions, a fixed partition policy was imposed. For such a policy, customers are partitioned into disjoint subsets or regions and each region is served separately and independently from the others. Although all the stores in a region were replenished together, direct shipments were allowed when circumstances required. The problem for each region was represented on a graph consisting of $\tau + \tau_0$ nodes and a set of directed edges, where τ denotes the number of periods in the planning horizon and τ_0 the maximum time allowed between successive deliveries. The nodes corresponded to delivery times and the edges to successive deliveries. The final T_0 nodes were included to ensure periodicity in the schedule.

An edge (t_1,t_2) represented successive deliveries to the region at times t_1 and t_2 with an associate cost of $c_e(t_1,t_2)$. The objective was to minimum total transportation costs of all shipments to a region, which could be achieved by finding the shortest path in the graph. Inventory holding costs were ignored because the requirement of frequent replenishment kept them low. Computational results showed that for the 207 stores in the Albert Heijn system, a 4% savings in distribution costs was achieved in the first year of implementation.

Irnich et al. (2006) introduced sequential search as a generic technique for the exploration of local neighborhoods and described when and how it can be incorporated in metaheuristics. They applied their sequential search algorithm to the capacitated vehicle routing problem (CVRP) and compared the result to lexicographic search approaches. The basic idea of sequential search is to consider all relevant partial moves of a cyclic independent neighborhood recursively. This accelerates the local search by pruning areas that are not likely to contain good candidates as early as possible so that only a small fraction of the entire neighborhood has to be scanned. Pruning is based on an evaluation of partial moves and their corresponding partial gains. The neighborhoods considered by

the authors included 3-opt, 2-opt, swap, relocation, or-opt and string-exchange. The computational experiments indicated that high quality solutions could be obtained for problem instances ranging from 250 to 2500 customers. Speedup factors of up to 10,000 were realized for the 3-opt neighborhood to cite one example.

Fukasawa et al. (2006) proposed a branch-and-cut-and-price for algorithm for the CVRP. They presented an alternative to the traditional Dantzig-Wolfe master problem that they called *Explicit Master*, which was based on the reformulation technique suggested by Poggi de Aragão and Uchoa (2003). The benefit of the *Explicit Master* formulation is that it does not change the structure of the subproblem when cuts are added to the master problem. In the methodology, framed capacity, strengthened comb, multistar, partial multistar, generalized multistar and hypotour inequalities were used to strengthen the formulation. The pricing subproblem was a constrained shortest path problem that consisted of finding q-routes of minimum reduced cost, where a q-route is a sequence of stops that starts at the depot, traverses a set of clients whose total demand does not exceed the vehicle capacity, and returns to the depot. Several heuristics were used to accelerate the solution of the subproblem. Testing showed that instances with up to 135 customers could be routinely solved with runtimes ranging from near 0 to 801963 seconds on a Pentium IV computer running at 2.4 GHz.

Parthanadee and Logendran (2006) considered a multi-product, multi-depot periodic distribution problem and formulated it as a MIP. In the model, they assumed that the daily demand of each customer was known and that all deliveries could be completed in one day within specified time windows and allowing for multiple vehicle trips. Backordering of products at the depots was allowed. To rationalize deliveries over the planning horizon, a set of predefined patterns was introduced and ranked by the customers. A penalty scheme was used to direct the search away from those that were deemed undesirable.

Three tabu search heuristics with different memory structures were proposed to find solutions. Initial solutions were constructed a cheapest insertion procedure, where each customer is assigned to the nearest depot and is given its most preferred delivery pattern. Trips were generated using a routing priority list that took into account the

customers' time windows. The result was that the customer who needed to receive a delivery the earliest was placed at the top of the list and so on. The search neighborhood was defined by depot and pattern delivery interchanges.

In the simplest version of the heuristic, short-term memory was implemented using tabu tenure only. Next, a diversification process was proposed that incorporated a penalty term in the objective function to account for undesirable attributes (delivery pattern assigned to customer or depot assigned to customer) associated with a solution. Finally, they developed an intensification process based on a long-term memory function that encourages solutions with good attributes. Performance was tested on various problem instances using a randomized complete block design. As expected, the results confirmed that the most sophisticated heuristic that used the diversification process was superior to the other implementations.

In their version of the PRP, Mourgaya and Venderbeck (2006) proposed a column generation-based heuristic to fix dates for customer visits and to assign customers to vehicles. The daily sequence decisions were left to an operational model. In formulating the problem two objectives were considered: (1) optimizing the compactness of the geographical regions to which customers were assigned, and (2) balancing the workload evenly between vehicles. Using a Dantzig-Wolfe reformulation, they found that the resulting master problem provided substantially stronger lower bounds than the LP relaxation of the original problem and that there were fewer difficulties due to symmetry during branch and bound. The pricing subproblem decomposed into τ clustering problems, one for each time period.

In the solution methodology, an initial set of feasible columns for the master problem was obtained with the construction heuristic of Mourgaya and Vanderbeck (2006). Because of the difficulty in solving the subproblems exactly, a primal insertion heuristic was used. Columns that priced out negatively were added to the master problem and the process repeated. When all subproblem objective function values were nonnegative, column generation ceased. At that point, a check for integrality was made and if none of the master problem variables were fractional, a feasible solution to the original problem was at hand. Otherwise, a rounding heuristic was called that fixed all

the master problem variables that were integral and rounded up the largest fractional variable. The column generation process was then repeated as many times as necessary to get a feasible solution. However, if the residual problem becomes too restricted, the remaining vehicles may not be sufficient to cover the required customer visits. In those cases, feasible solutions were obtained with the construction heuristic by exploring different insertion strategies and selecting the best available assignments.

To speed convergence during column generation, an attempt was made to find a lower bound on the master problem. To get the tightest bound, it was necessary to solve the subproblems exactly, something that was only possible for instances with up to 20 customers. Alternatively, a lower bound on each subproblem was computed by dualizing its constraints and using a hybrid subgradient procedure to find solutions. Unfortunately, the resultant bounds proved too weak to be of any value.

Computational tests were performed using 20, 50 and 80 customer data sets that involved comparisons of various implementation options related to initialization, column generation strategies, number of passes in the rounding heuristic, and problem specifications. The results showed that adding extra columns at the beginning obtained from other feasibility heuristics did not bring any benefit on average. With regard to column generation strategies, it was observed that adding multiple columns at each iteration rather than only the first negatively priced column found, improved solution quality by 6.4% on average without increasing runtimes. However, when adding too many columns from each subproblem, they observed no improvement in solution quality but larger runtimes. The results also showed that by using three passes instead of one for rounding heuristic led to a 13.80% improvement in solution quality on average.

Zhao et al. (2007) studied the integration of inventory control and vehicle routing for a distribution system in which a set of retailers with constant rates of demand were resupplied with a single item from a central warehouse. The objective was to determine inventory policies and routing strategies such that the long-run average costs were minimized and all demands were satisfied. In their model, no inventory capacity constraints were imposed on the warehouse or on the retailers. Transportation cost consisted of a fixed cost plus a variable cost proportional to the total distance traveled.

Inventory holding costs were assessed at the retailer sites and the warehouse assuming a constant rate per item per unit of time. A fixed ordering cost was also incurred each time inventory at the warehouse was replenished.

Next, a lower bound for the long-run average cost of any feasible strategy was determined. The costs considered in the model included the average of the ordering cost (c_1) and the holding cost at the warehouse (c_2), the average of the holding cost at the retailers (c_3), and the average of the transportation cost incurred by the vehicles traveling from the warehouse to the retailers (c_4). The lower bound on $c_1 + c_2 + c_3$ was obtained by relaxing the restriction on the capacity of the vehicles while the lower bound on c_4 was derived from the procedure given by Chan et al. (1998). The authors also analyzed the relationship between the average cost of the optimal feasible solution and the proposed lower bound, and used it to assess the quality of the computational results.

To simplify the problem, a fixed partition policy was used in which the retailers were grouped into separate, nonoverlapping regions. A tabu search algorithm was developed for this purpose and the *power of two* (POT) principle (Roundy 1985) was used to find the replenishment intervals for each region and the warehouse. To determine the POT intervals, the average cost of the system was first expressed for a given decomposed set of retailers and a corresponding set of the POT replenishment intervals for each set of retailers. Next, the lower bound on the average cost was determined by using the optimal replenishment interval in each region based on the EOQ formula. The authors showed that this new function is convex and differentiable except for the break points -- the points where the function changes. Using those break points, they derived the POT replenishment intervals by following the method given by Anily and Federgruen (1993).

Testing was done on problem instances with 50 and 75 retailers. By analyzing the relationship between the lower bound and the optimal solution, they were able to establish that the proposed algorithm in conjunction with the fixed partitioning policy were effective and robust in almost all cases.

2.3 PRODUCTION PLANNING AND INVENTORY

There has been a vast amount of research in the areas of production planning and inventory management over the last 40 years. Much of this work relates to lot sizing as described in Section 2.1, and most models presented to date treat production and distribution separately. Beginning with Clark and Scarf's (1960) analysis of multi-echelon inventory systems, the field has grown to include all aspects of the supply chain. Suppliers are the start of the chain providing raw material to production sites. Finished goods are stored on site or at warehouses before being distributed to customers. A common objective is to minimize the total costs associated with production, holding, and distribution. If only a central inventory system (warehouse) exists, then the problem becomes an inventory-distribution problem. If production facilities are considered as well, then the problem becomes a production-inventory-distribution problem. As mentioned, the routing decision is part of the PIDRP but in the more general case, it may only necessary to take into account the distribution mode and timing rather than the specific routing.

It is not our intention to review all related production-inventory literature here. The focus is on the models that involve either coordination between inventory and distribution or between production, inventory and distribution.

Glover et al. (1979) investigated a production and distribution problem that arose at Agrico Chemical by formulating it as network flow model. Their approach was motivated by the fact that such models are readily solvable, even for large instances. The primary aim of the project was to examine strategic decisions related to facility location, the amount of long-term inventory investment, the number of distribution centers, and the size of transportation equipment. Solutions were obtained with a combination of software packages called ARCNET and PNET/LP for problem instances that typically contained 6250 constraints and 23,000 variables.

For a multi-plant, multi-product model with identical vehicles, Chandra and Fisher (1994) investigated the value of coordinating production and distribution by comparing two different solution strategies. The first was a two-stage approach in which the production schedule was determined by solving a capacitated lot-sizing problem that

aggregated all customer demand in each period. Next, the actual routings were determined for each period with common heuristics such as sweep, nearest neighbor rule, and a feasible insertion rule. In a post-processing step, the solution was improved by combining two or more deliveries to a customer on the same day; however, the production schedule was not allowed to change during this step.

In the second approach, the production scheduling and vehicle routing were coordinated within a single model. The initial production and distribution schedule were determined by the same procedure used in the first approach. In the second stage, a search was made for additional cost-reducing changes and the production schedule is allowed to change. These included the consolidation of deliveries to each customer between different periods and changes in the production schedule when moving a delivery from one period to another. When the production schedule is allowed to change, the inventory availability constraint becomes a production capacity constraint. In either case, the authors solved problem instances with up to 50 customers, 10 periods, and 10 products. The computational results showed that by using the coordinated approach, the total cost could be reduced by 3 to 20%, depending on the values of the structural parameters associated with the length of the planning horizon, the number of products, the number of customers, and the production capacity.

Anily and Federgruen (1990) analyzed fixed partition policies for the inventory routing problem with constant deterministic demand rates and an unlimited number of vehicles. The routing patterns were determined by using a modified circular partition scheme. After the customers were partitioned, those within a partition were assigned to one or more regions. Demand was calculated by summing the demand of the individual customers assigned to a region. When a customer appeared in multiple regions a percentage of his demand was allocated to each. The routing logic required that all customers in a region be visited as long as there was a need to visit one. A lower bound on the long-run average cost was also determined to provide an understanding of the effectiveness of the routing scheme.

Anily and Federgruen (1993) extended their earlier work to include the case where the depot can hold inventory. A new algorithm in which the interval between two

visits to a region follows the POT principle was developed. They showed that the different between the cost from this algorithm and a lower bound on the minimum cost obtained by similar approach proposed by Roundy (1985) for uncapacitated systems was at most 6 % for a sufficiently large number of retailers.

Herer and Roundy (1997) investigated the one warehouse multi-retailer distribution problem where the vehicle routing cost was based on the length of the traveling salesman tour beginning and ending at the central warehouse and visiting a subset of retailers. In the model, they assumed that a central warehouse receives deliveries from an outside supplier(s), and that inventories can be held at both the warehouse and at the retailer sites. Demand was specified for the retailers only and was assumed to be deterministic. The objective was to minimize the long-run average costs, which consisted of fixed ordering costs and inventory holding costs. The ordering costs were assumed to be the sum of two cost functions, where the first was given as a monotone nonnegative submodular function of the set of retailers placing orders at a given point of time. Included in this category were the warehouse ordering costs, the loading and unloading costs at the retailer sites, the processing costs at the retailer sites, and the truck rental costs. The second function took into account transportation costs, which were assumed to be proportional to the distance traveled by a delivery vehicle. Inventory holding costs were the same as those defined for the standard EOQ model.

The authors developed a number of polynomial time heuristics, which produced results that were provably close to the cost of an optimal policy. They showed that given a submodular function that is a good approximation of the true ordering cost, there exists a POT policy whose cost is only moderately greater than the cost of an optimal policy. They also developed a nonpolynomial-time dynamic program that computes optimal POT policies for a one warehouse multi-retailer system assuming that the ordering costs are arbitrary nonnegative monotone in number of items ordered. In the computational testing, a comparison was made between the heuristics and the optimal POT policies for instances with up to 16 retailers. The results demonstrated that the ratio of the cost of the POT policy calculated by the heuristics to the cost of an optimal policy was bounded by a

factor that depended on the closeness of the traveling salesman tour length to its submodularity approximation.

Fumero and Vercellis (1999) approached the PIDRP by proposing an integrated optimization model that considered the production and distribution of multiple products under limited plant capacity and fleet availability. Two solution approaches were compared. The first was based on a decoupling procedure similar to the one developed by Chandra and Fisher (1994). The other was a synchronized approach based on Lagrangian relaxation to decompose the integrated model into four separate subproblems. A global optimization perspective was preserved through the dual master problem. The results for problem instances with 8 – 12 customers, 5 – 8 periods, and 5 – 10 products showed that the synchronized approach provided a substantial advantage over the decoupled decision process.

Özdamar and Yazgac (1999) proposed a production-distribution model that considered production and transportation decisions at a central factory and its warehouses. The model was based on the operations of a multi-national company producing detergents. In the study, the objective was to develop a scheduling plan that minimized production, inventory and transportation costs. Constraints included production capacity, inventory balance, and fleet size limits. In the hierarchical approach proposed, an aggregate planning model that reflected the system's physical constraints was first solved to obtain a rough operational plan over the planning horizon. Although the number of vehicles was optimized, no routing information was provided at this stage.

In the high level model, products, demand, capacity, and time periods were aggregated across product families. Rather than addressing capacity consumption by setups directly, an approximate percentage of capacity was allocated to setups. Next, the solution from the aggregate model was refined in terms of time periods, product families, inventory, and distribution quantities by solving the disaggregated model that is a MIP repeatedly until a feasible solution is obtained. To resolve inconsistency such as undesirable back order levels or insufficient production capacity imposed by aggregate quantities that might have resulted from the aggregated solution, it was necessary to modify or remove relevant constraints in the disaggregated models. They also

investigated the robustness of the hierarchical model in terms of infeasibilities occurring due to the highly fluctuating nature of demand from one time period to the next. Testing was done on problem instances defined by 6 periods, 2 products and 5 warehouses.

Van Buer et al. (1999) developed several heuristics for solving a newspaper production and distribution problem. Neither inventory nor production costs were considered in their model. The focus was on distribution for a single day. Tabu search, reactive tabu search, and simulated annealing with different sets of cooling parameters were compared on medium size problems with 30-100 distribution centers and carrier routes consisting of up to 200 home subscribers. Testing showed no significant performance difference among the search algorithms.

Jayaraman and Pirkul (2001) proposed an integrated logistics model for locating production and distribution facilities in a multi-echelon environment. In the model, they considered two levels of decisions, one strategic (where to locate plants and warehouses) and the other operational (how do distribute their products from the plants to customer outlets through warehouses). They provided a MIP formulation for the integrated model and used Lagrangian relaxation to derive bounds. A heuristic procedure based on linear programming techniques coupled with the results from the Lagrangian relaxation was used to generate feasible solutions. This involved three steps: first, customers were assigned to warehouses; next, a production levels were determined; and finally, the distribution problem was solved. The computational results showed that the procedure produced high quality solutions within acceptable runtimes. For large instances with 75 customers, 15 possible warehouse locations, 10 possible plant locations, 3 products, and 2 suppliers, optimality gaps ranged from 0.05 to 2.86%.

Bertazzi et al. (2002) considered a deterministic version of the IRP with a single capacitated vehicle over a long-term period. The objective was to minimize the cost of deliveries to the retailers while ensuring that inventory levels were maintained between some minimum and maximum values. They presented a two-phase heuristic to determine the vehicle route at each discrete time point while using an order-up-to policy to management inventory. In the initialization phase, a feasible solution is built with a sequential procedure that inserts a retailer at each iteration. When a retailer is being

considered, a set of delivery time instants is determined by solving a shortest-path problem. Then, for each of the selected delivery time instants, the retailer is inserted into the route at the point that minimizes the incremental cost.

In the second phase, the incumbent is improved by performing swaps that removed or inserted retailers at different positions of the route used by the vehicle. They compared the solution from their two-phase proposed algorithm with the optimal cost of two intuitive policies, *Every* and *Latest*. *Every* visits all retailers on the basis of an optimal route based on traveling distance. *Latest* visits the set of retailers based on the stockout condition in which the set of retailers will have stockout if not served at the current time. Testing on a set of randomly generated problem instances with 50 retailers and 30 planning periods showed that the solution obtained by the proposed algorithm outperformed the solutions provided by the two intuitive policies.

Lei et al. (2006) proposed a two-phase solution approach for the PIDRP that permitted a direct shipment strategy. In phase one, a restricted version of the problem that contained all but the routing constraints was solved. The results provided a production schedule and the number of items to be delivered to each customer in each period. The solutions were shown to always be feasible to the original problem. However, they did not allow for the consolidation of less-than transporter loads (LTL), which could have reduced overall transportation costs. In phase two, a routing heuristic was proposed based on an extended optimal partitioning procedure that consolidated the LTL assignments obtained in phase one into more efficient delivery schedules. The two-phase method successfully solved problem instances with up to 50 distribution center locations over 2 to 4 planning periods.

Boudia et al. (2006, 2007) proposed both a memetic algorithm with population management (MAPA) and a reactive greedy randomized adaptive search procedure (GRASP) with path-relinking (Prais and Ribeiro 2000, Resende and Ribeiro 2005) to solve the PIDRP. The model included a single plant and a set of customers located on a grid. Holding costs at the customer sites were assumed to be negligible compared to the holding costs at the plant and so were ignored. The objective was to minimize the sum of

production, holding and transportation costs while ensuring that all demand was satisfied over the planning horizon.

Like most metaheuristics, GRASP has two phases. In the construction phase, a preliminary production plan and a set of delivery schedules were created and then post-processed by a Wagner and Whitin-type algorithm to find a better balance between setup and holding costs. In the local search phase, two neighborhoods were defined for further exploration. For the first, all pairs of customers who were visited on the same day were considered eligible for swaps. For the second, products delivered to a customer on day t were allowed to be moved to any other day as long as all constraints were respected.

Path relinking is aimed at exploring the paths between elite solutions found during the computations. The motivation for type of analysis comes from the topological idea that a valley, and hence an improved solution, may exist between pairs of feasible points. To limit the search to high quality candidates, the authors created a pool of elite solutions by saving a predetermined number of new GRASP incumbents. The resulting pool was sorted in increasing order of objective function costs and updated during successive GRASP iterations. Two techniques were proposed for updating the procedure: (1) replace the last member of the pool each time a better solution is found and (2) use the technique suggested by Resende and Ribeiro (2005) to increase pool diversity. Next, all pairs of elite solutions were examined for path relinking. Given one initiating solution A and one guiding solution B the path between them was generated by modifying progressively A to reproduce in each day t the trips present in B.

As an enhancement, the authors proposed two strategies to combine path relinking and GRASP. The first strategy was to perform path relinking upon the termination of GRASP; the second was to perform path relinking every time GRASP provided a new incumbent. The computational results showed that on average the first strategy performed better than the second on the 50 and 100 customer instances with 20 time periods but was not as good on the 200 customer instances. The algorithms converged within 100 sec on a 2.8 GHz computer in all cases but no lower bounding procedures were provided to gauge the quality of the solutions. The reactive GRASP on average

provided more savings than the solutions obtained with the GRASP alone by 0.8% and the solutions obtained with the relinking feature by 0.42%.

A MAPA is a genetic algorithm in which a mutation is replaced by a diversification mechanism based on a distance measure in the solution space. Using an MAPA for the PIDRP, Boudia et al. (2006) designed a two-part chromosome to encode and manipulate a solution. The first part is a vector of integers in which each element represents the amount produced in each period. The second part is a list consisting of trips in successive periods. The fitness of a chromosome is the total cost of the associated solution. The initial population was generated by randomly choosing the number of production days and then using the Clarke and Wright savings heuristic to build trips. The parents for a crossover were chosen with the binary tournament method and the parameters were set so that the crossover operator generated only one child. The local search component of the algorithm was similar to the one used in the GRASP. Testing showed that the MAPA increased saving by 3% on average compared to the results obtained with the reactive GRASP.

Cetinkaya et al. (2006) investigated the impact of alternative outbound dispatch policies on integrated stock replenishment and transportation decisions in a VMI system. In their model, retailers were grouped based on geographic region and it was assumed that the replenishment costs were fixed. Demand was generated randomly for each group and deliveries were made in batches of consolidated loads. In the analysis, they compared the performance of time-based and quantity-based policies. Generally speaking, a time-based policy ships accumulated loads every fixed T periods whereas a quantity based policy ships accumulated loads when a specific dispatch quantity q is reached.

After preliminary testing of each policy separately, a hybrid dispatching policy was proposed that tried to consolidate outbound loads of size q_H. However, if the time since the last dispatch exceeded T_H before the dispatch quantity q_H accumulated, then the dispatch was made immediately. Several rules were examined for setting the parameters q_H and T_H. The results showed that the hybrid policy was superior to the time-based policy alone on all measures, but not to the quantity-based policies with respect to cost.

When long-run average waiting times were used as a service measure, though, the hybrid policy won out.

Chapter 3

Model Formulation

We are given a single production facility and a set of n customers geographically dispersed on a grid. Each customer i has a fixed nonnegative demand d_{it} in time period t of the planning horizon that must be satisfied; i.e., shortages are not permitted. If production takes place at the facility in period t, then a setup cost f_t is incurred, $t = 0,1,\ldots,\tau$. A limited number of items can be produced in each time period and a limited number can store at a unit cost of h^P. In general, it is natural to equate a period with a day but when production is scheduled for more than one shift in a day, a broader interpretation of a period is appropriate.

In constructing delivery schedules, each customer can be visited at most once per day and each of θ homogeneous vehicles can make at most one trip per day. The latter restriction implies that all routes overlap in time. A limited amount of inventory can be stored at customer i's site at a unit cost of h_i^C, but transshipments between customers are not permitted.

Moreover, it is assumed that all deliveries take place at the beginning of the day and arrive in time to satisfy demand for at least that day. All production on day t is available for delivery only on the following morning (this is common in food production and distribution; e.g., see Villegas and Smith 2006) and all inventory is measured at the end of the day. Demand on day t can be met out of deliveries from the factory on day t and ending inventory on day $t - 1$ at the customer site. Initial customer inventory on day 0 simply reduces demand on subsequent days, while initial inventory at the factory must be routed; at the end of the planning horizon all inventory is required to be zero.

The objective is to construct a production plan and delivery schedule that minimize the sum of all costs while ensuring that each customer's demand is met over the planning horizon. Implicit in the solution is a daily distribution of surplus items to be

placed in inventory. Under the common assumption that unit production costs are constant, the decision to overproduce and hold items in inventory is a function of the setup cost at the factory, system holding costs, production and storage capacities, vehicle routes, and daily demand.

In the development of the model, we make use of the following notation.

Indices and sets

i, j	indices for customers, where 0 corresponds to the production facility
t	index for periods or days
N	set of customers; $N_0 = N \cup \{0\}$; $\|N\| = n$
T	set of days in the planning horizon; $T_0 = T \cup \{0\}$ and $\|T\| = \tau$

Parameters

d_{it}	demand of customer i on day t
θ	number of available vehicles
Q	capacity of each vehicle
$I^P_{\max}$	maximum inventory that can be held at the production facility
$I^C_{\max.i}$	maximum inventory that can be held by customer i
C	production capacity of the factory
c_{ij}	cost of going from customer i to customer j
f_t	setup cost at production facility on day t
h^P	unit holding cost at the production facility
h^C_i	unit holding cost at customer i site

Decision variables

x_{ijt}	1 if customer i immediately precedes customer j on a delivery route on day t; 0 otherwise
y_{it}	load on a vehicle immediately before making a delivery to customer i on day t
p_t	production quantity on day t
z_t	1 if there is production on day t; 0 otherwise
I^P_t	inventory at the production facility at the end of day t

I_{it}^C inventory at customer i at the end of day t

w_{it} amount delivered to customer i on day t

Model

$$\phi_{\text{IP}} = \text{Minimize} \quad \sum_{t\in T}\sum_{i\in N_0}\sum_{j\in N_0} c_{ij}x_{ijt} + \sum_{t\in T} f_t z_t + \sum_{t\in T_0\setminus\{\tau\}} h^P I_t^P + \sum_{t\in T\setminus\{\tau\}}\sum_{i\in N} h_i^C I_{it}^C \tag{3-1a}$$

$$\text{subject to} \quad I_t^P = I_{t-1}^P + p_t - \sum_{i\in N} w_{it}\,, \ \forall\, t \in T_0 \tag{3-1b}$$

$$I_{it}^C = I_{i,t-1}^C + w_{it} - d_{it}\,, \ \forall\, i \in N, t \in T \tag{3-1c}$$

$$\sum_{i\in N} w_{it} \le I_{t-1}^P, \ \forall\, t \in T \tag{3-1d}$$

$$p_t \le Cz_t, \ \forall\, t \in T_0 \setminus \{\tau\} \tag{3-1e}$$

$$p_0 \ge \sum_{i\in N}\left(d_{i1} - I_{i0}^C\right) \tag{3-1f}$$

$$\sum_{\substack{j\in N_0\\ j\ne i}} x_{ijt} \le 1\,, \ \forall\, i \in N, t \in T \tag{3-1g}$$

$$\sum_{\substack{i\in N_0\\ i\ne j}} x_{ijt} = \sum_{\substack{i\in N_0\\ i\ne j}} x_{jit}\,, \ \forall\, j \in N, t \in T \tag{3-1h}$$

$$\sum_{j\in N} x_{0jt} \le \theta\,, \ \forall\, t \in T \tag{3-1i}$$

$$y_{jt} \le y_{it} - w_{it} + D_t^{\max}(1 - x_{ijt}), \ \forall\, i \in N, j \in N_0, t \in T \tag{3-1j}$$

$$w_{it} \le D_{it}^{\max} \sum_{j\in N_0} x_{ijt}\,, \ \forall\, i \in N, t \in T \tag{3-1k}$$

$$0 \le I_t^P \le I_{\max}^P\,,\ 0 \le I_{it}^C \le I_{\max,i}^C\,, \ \forall\, i \in N, t \in T\setminus\{\tau\};\ I_\tau^P = I_{i\tau}^C = 0\,, \forall\, i \in N \tag{3-1l}$$

$$x_{ijt} \in \{0, 1\}, z_t \in \{0, 1\}, 0 \le y_{it} \le Q, \ p_t \ge 0, \ w_{it} \ge 0 \text{ and integer}, \forall\, i \ne j \in N_0, t \in T \tag{3-1m}$$

where $D_{it}^{\max} = \min\left\{Q, \sum_{l=t}^{\tau} d_{il}\right\}$ and $D_t^{\max} = \min\left\{Q, \sum_{i\in N}\sum_{l=t}^{\tau} d_{il}\right\}$

The objective function minimizes the sum of transportation costs, production setup costs, holding costs at the factory, and holding costs at the customer sites. Because all demand must be met, production costs, which are assumed to be linear and constant, can be omitted, as can any initial inventory costs. Constraints (3-1b) and (3-1c) are inventory flow balance equations in which it is assumed that the initial inventories I_0^P and I_{i0}^C are given for all customers $i \in N$. Constraint (3-1e) limits production on day t to the capacity of the factory. A simple way to tighten this constraint is to replace C with $C_t = \min\left\{C, I_{\max}^P, D_{t+1}^{\max}\right\}$, where the third term is the demand of all customers for the remainder of the planning horizon. The assumption that items produced on day t are only available for delivery on day $t+1$ implies that $p_t \le I_{\max}^P$ and $p_\tau = 0$. It is possible to strengthen the latter inequality by subtracting from $I_{\max}^P$ the reduction in inventory due to deliveries on day t to get $p_t \le I_{\max}^P - \left(I_{t-1}^P - \sum_{i \in N} w_{it}\right)$, but this constraint is dominated by (3-1b). To ensure that demand on day 1 can be met, it is necessary to include (3-1f) which allows production on day 0. If $I_0^P = I_{i0}^C = 0$, then $p_0 \ge \sum_{i \in N} d_{i1}$ or the problem is infeasible.

As indicated by constraint (3-1d), the total amount available for delivery on day t is limited by the amount in inventory at the factory on day $t - 1$. The specific amount delivered to customer i is limited by the parameter D_{it} in (3-1k), which is the smaller of the vehicle capacity Q or the cumulative demand from day t to the end of planning horizon τ. When customer i has no successors on a route, no delivery is possible because the right-hand side of (3-1k) would be zero.

Constraints (3-1g) – (3-1j) represent the routing aspect of the problem. Constraint (3-1g) ensures that if customer i is serviced on day t, then it must have a successor on its route, which may be the factory. Route continuity is enforced by (3-1h); i.e., if a vehicle arrives at customer j on day t, then a vehicle must depart customer j on day t. Logic, and the fact that the fleet is homogeneous, requires that it be the same vehicle.

The number of vehicles that leave the factory on any day t is limited to the number available, θ, as indicated by (3-1i). Finally, (3-1j) keeps track of the load on the vehicles and guarantees that on day t, if customer i is the immediate predecessor of customer j on a route, then the load on the vehicle before visiting customer j must be less than or equal to the load just before visiting customer i minus the amount delivered, which is represented by the variable w_{it}. The value of the parameter $D_t^{\max}$ is specified to be as small as possible while ensuring that (3-1j) is feasible in all cases. Because the load on each vehicle is monotonically decreasing as customers are visited, (3-1j) provides the added benefit of eliminating subtours that do not include the factory. After all deliveries are made, the fleet returns to the factory so y_{j0} can be set to 0 for all $j \in N$. To conclude the formulation, variable bounds are specified in (3-1*l*) and (3-1m).

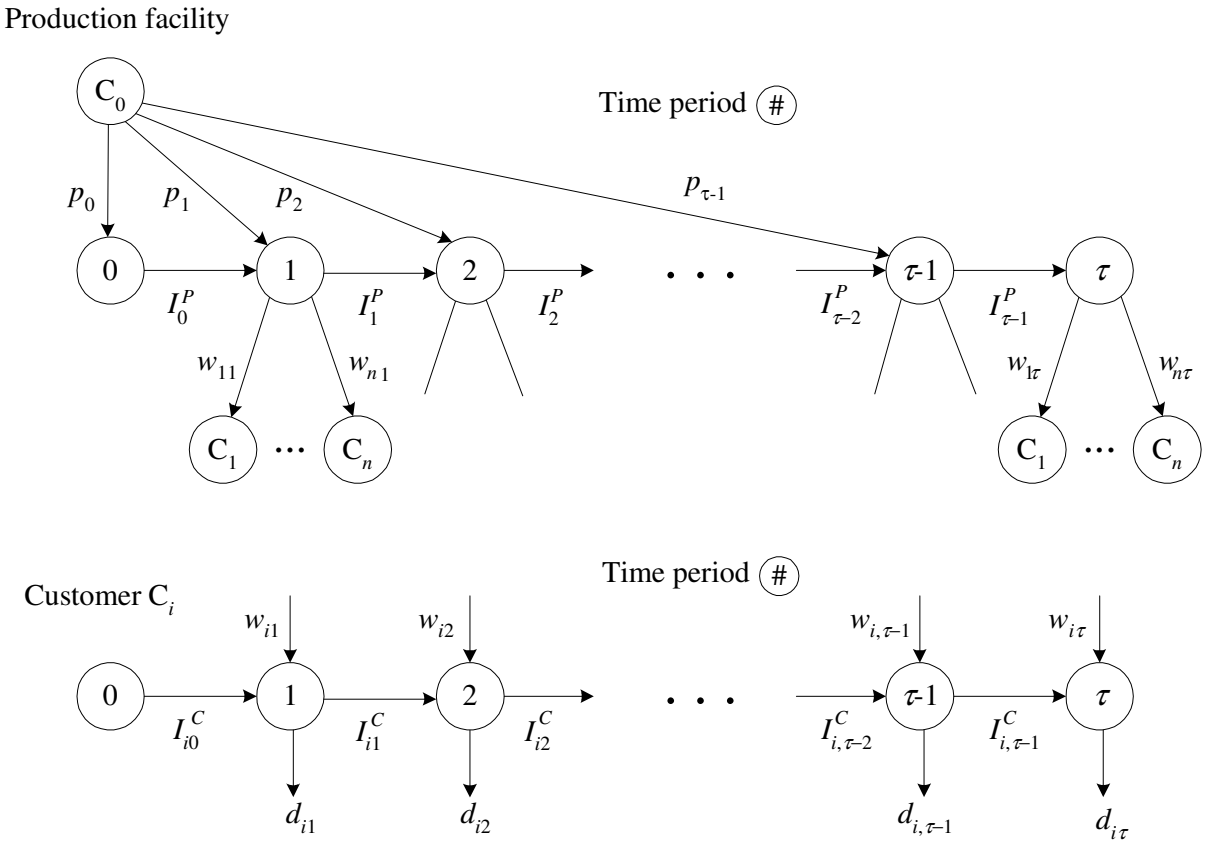

Figure 3.1. Network flow representation of production-distribution problem

Figure 3.1 depicts the network flow diagram for our extended single-facility, single item lot-sizing problem with routing considerations. The top portion of the figure represents the production facility, denoted by C_0, for days 0 through τ. Instead of demand driving the decisions on each day t for the factory, a series of delivery decisions has to be made for each customer i. The amounted delivered, denoted by w_{it}, is combined with customer i's inventory $I^C_{i,t-1}$ at the beginning of day t to satisfy demand, d_{it}. The corresponding flow is shown in the bottom portion of the figure. Because the inventory at the end of the planning horizon at both the factory and the customers' sites is required to be zero, there is no horizontal flow exiting node τ in either network.

The size of model (3-1) is determined largely by constraint (3-1j) and the number of binary variables, x_{ijt}, both of which grow at a rate proportional to $O(n^2 \tau)$. The majority of the other constraints only grow at a rate proportional to $O(n\tau)$. A small problem with 3 vehicles, 30 customers, and a 5-day planning horizon contains roughly 5200 constraints, 4600 binary variables, and 500 continuous variables. Initial attempts to solve instances of this size with CPLEX 8.1 were not encouraging. When a time limit of 90 minutes was imposed, optimality gaps between 7% and 10% were the norm.

This experience led to development of a tabu search algorithm (Glover 1989, and Glover 1990) presented in the next section and an exact method based on branch and price (B&P) presented in Section 7. Additional algorithmic components include an initialization scheme, heuristics for generating feasible solutions, and perhaps approximation methods for balancing computational efficiency with solution quality are presented in Section 9.

Chapter 4

Solution Methodology – Tabu Search

A three-phase approach is used in the design of our tabu search algorithm for solving the PIDRP In phase 1, an initial solution is found by solving a simplified version of model (3-1) in which the routing variables (x_{ijt}) and constraints (3-1g) – (3-1j) are removed and aggregate vehicle capacity constraints are added. We call this the *allocation model*; its solution provides customer allocations w_{it} for all $i = 1,\ldots,n$ and $t = 1,\ldots,\tau$. In phase 2, the results of phase 1 are used to solve τ independent routing problems. An efficient CVRP subroutine based on tabu search (Carlton and Barnes 1996) is used for this purpose. In phase 3, neighborhood search is performed to improve the allocations and routing assignments found in phase 2.

4.1 INITIAL SOLUTION

To find initial solutions, we modify model (3-1) to create a pure lot-sizing distribution problem without a routing component and then solve it as a MIP to get a set of feasible allocations for each day in the planning horizon. The formulation requires the following additional notation.

Additional parameters

f_{it}^C fixed cost of making a delivery to customer i on day t

e_{it}^C variable cost of delivering one item to customer i on day t

Additional decision variable

z_{it}^C 1 if a delivery is made to customer i on day t; 0 otherwise

Because the actual cost of making a delivery to customer i on day t cannot be determined without including the routing constraints, an alternative representation is

needed for the cost term $\sum_{ijt} c_{ij} x_{ijt}$ in model (3-1). For problem instances with $n^2\tau \leq$ 500, we use the surrogate cost term $\sum_{it} f_{it}^C z_{it}^C$ with $f_{it}^C = 2c_{i0}$ for all *i* and *t*. In another words, the routing costs on any day *t* are approximated by the cost of a round trip between the depot and customer *i* (the model solution depends on the relative value of the holding costs and the approximate transportation cost, in this case it got to be the round trip value $2c_{i0}$). The variable cost e_{it}^C is set to zero. When $n^2\tau > 500$, indicating a fairly large instance, it is not practical to solve the *allocation model* given below with the setup variables z_{it}^C included. In this case, we set $z_{it}^C = f_{it}^C = 0$ for all *i* and *t*, and use the variable cost term $\sum_{it} e_{it}^C w_{it}$ to replace $\sum_{ijt} c_{ij} x_{ijt}$, where e_{it}^C is approximated by the cost of making a delivery to customer *i* directly from the depot divided by the total demand of customer *i* on day *t*; i.e., $e_{it}^C = 2c_{0i} / d_{it}$.

Allocation Model

$$\phi_{\text{IP}} = \text{Minimize} \quad \sum_{t\in T} f_t z_t + \sum_{t\in T}\sum_{i\in N} f_{it}^C z_{it}^C + \sum_{t\in T}\sum_{i\in N} e_{it}^C w_{it} + \sum_{t\in T_0\setminus\{\tau\}} h^P I_t^P + \sum_{t\in T\setminus\{\tau\}}\sum_{i\in N} h_i^C I_{it}^C \tag{4-1a}$$

$$\text{subject to} \quad I_t^P = I_{t-1}^P + p_t - \sum_{i\in N} w_{it}, \ \forall\, t \in T_0 \tag{4-1b}$$

$$\sum_{i\in N} w_{it} \leq I_{t-1}^P, \ \forall\, t \in T \tag{4-1c}$$

$$p_t \leq C z_t, \ \forall\, t \in T_0 \setminus \{\tau\} \tag{4-1d}$$

$$p_0 \geq \sum_{i\in N} \left(d_{i1} - I_{i0}^C\right) \tag{4-1e}$$

$$I_{it}^C = I_{i,t-1}^C + w_{it} - d_{it}, \ \forall\, i \in N, t \in T \tag{4-1f}$$

$$w_{it} \leq D_{it}^{\max} z_{it}^C, \ \forall\, i \in N, t \in T \tag{4-1g}$$

$$\sum_{i\in N} w_{it} \leq 0.8Q\theta, \ \forall\, t \in T \tag{4-1h}$$

$$0 \leq I_t^P \leq I_{\max}^P, \ 0 \leq I_{it}^C \leq I_{\max,i}^C, \ \forall\, i \in N, t \in T\setminus\{\tau\};$$

$$I_\tau^P = I_{i\tau}^C = 0, \ \forall\, i \in N \tag{4-1i}$$

$$z_t \in \{0, 1\},\ z_{it}^C \in \{0,1\},\ p_t \geq 0,\ w_{it} \geq 0 \text{ and integer}, \forall\ i \in N, t \in T \qquad \text{(4-1j)}$$

The primary difference between models (3-1) and (4-1) is that the routing variables (x_{ijt}) and associated constraints have been removed, which allows us to solve relatively large instances quickly. An additional variable z_{it}^C is included in model (4-1) to keep track of whether customer *i* receives a delivery on day *t*. Also, notice that a new set of constraints (4-1h) has been introduced. These constraints limit the total amount that can be delivered on day *t* to a fixed percentage of the total transportation capacity, and provides a hedge against the need for split deliveries. Testing showed that using a value of 80% always yielded feasible solutions.

As mentioned, the original transportation costs in (3-1a), are approximated in (4-1a) by using a delivery setup cost f_{it}^C in conjunction with a variable unit delivery cost e_{it}^C for each customer *i* and day *t*. Empirically, solutions to the allocation model were seen to be close to those of the full model when the production and fleet capacities were larger than the total daily demand. In other cases, it was necessary to use neighborhood search to improve upon the results.

Given a solution to model (4-1), an initial solution to the PIDRP is constructed with the algorithm outlined in Figure 4.1.

Although not specified in (4-1i) and (4-1j), our solution methodology requires that the delivery quantities w_{it} to be integral. The following proposition shows that the allocation model always returns integer values for not only w_{it}, but for p_t, I_t^P and I_{it}^C as well.

Solve the allocation model (4-1a) – (4-1j) with a standard MIP code (CPLEX is used in the implementation) to get $\left(\overline{p}_t, \overline{z}_t, \overline{z}_{it}^C, \overline{w}_{it}, \overline{I}_t^P, \overline{I}_{it}^C\right)$

For (all days $t \in T$) {

Use the amount delivered to each customer on day t $\left\{\overline{w}_{it} > 0 : i \in N\right\}$ as input to the capacitated vehicle routing subroutine. Determine the delivery routes for each day t.

}

Figure 4.1. Algorithm for constructing an initial solution

Proposition 4.1 When the setup variables z_t and z_{it}^C are fixed at 0 or 1, there exists and optimal solution to model (3-1a) – (3-1j) such that w_{it}, p_t, I_t^P and I_{it}^C are integral for all $i \in N$ and $t \in T_0$.

Proof. We show that when z_t and z_{it}^C are fixed, the remaining components of the model are equivalent to a pure network flow problem whose constraint matrix is known to be totally unimodular. The first step is to represent the flows at the plant more accurately by accounting for the fact that production on day t is held in inventory that day and is only available for delivery on day $t + 1$. This can be done by creating three inventory nodes at the plant in each period: one for serving customers, a second to bound the flow to customers, and a third to receive the current day's production. This division is shown in Figure 4.2, where primes and double primes are used to distinguish the three nodes. At node 1', for example, items may be withdrawn and delivered to customers; however, an upper bound of $\lfloor 0.8\theta Q \rfloor$ is placed on the arc from node 1' to node 1" to ensure adherence to the capacity restrictions (not shown). Items produced on day 1 are channeled to node 1 rather than node 1'. At the end of the day, whatever inventory remains flows to node 2', as indicated by the variable $I_1^{P'}$.

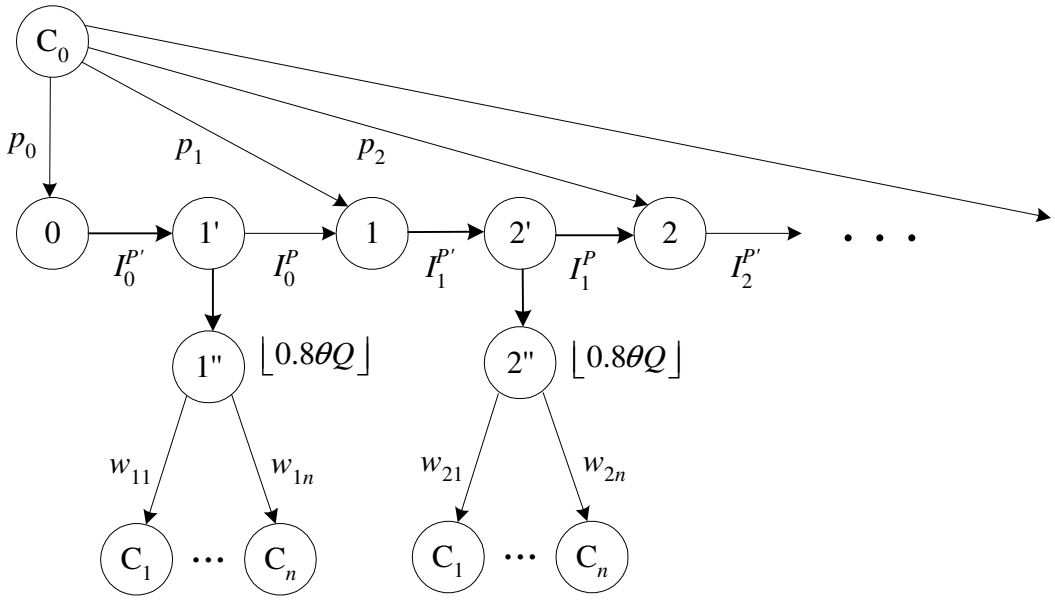

Figure 4.2. Network flow with detailed inventory nodes

When z_t and z_{it}^C are fixed, the remaining variables are bounded by integer values. Conservation of flow at the nodes with primes is $I_{t-1}^P = I_{t-1}^{P'} - \sum_{i \in N} w_{it}$, which is equivalent to (4-1c) given the variable bound $I_{t-1}^P \geq 0$ for all t. Conservation of flow at the nodes without primes is essentially Eq. (4-1b) with I_{t-1}^P replaced with its equivalent. The delivery limits in (4-1h) are enforced by the bounds on the arcs between the nodes with single and double primes. Finally, constraint (4-1f) represents conservation of flow at the customer sites and is already in pure network form. Thus we have a bounded pure network flow problem that is guaranteed to have an optimal integral solution. ■

4.2 SOLUTION REPRESENTATION

The solution is represented by the amount of production in each day t ($\bar{p}_t$, $t = 0,1,...,\tau$), and the quantity delivered to each customer i on each day t ($\bar{w}_{it}$, $t = 0,1...,\tau$; $i = 1,\ldots,n$).

4.3 NEIGHBORHOOD DEFINITION

A neighborhood is a set of points that can be reached from the current solution by performing one or more moves that, in general, take the form of insertions, exchanges or replacements. From any incumbent, a new solution is determined by identifying the best solution within its neighborhood. For the PIDRP, we define the neighborhood as all feasible points that can be reached by two types of moves between periods. The first is called a *swap* and involves a (partial) exchange of delivery quantities between two customers i_1 in period t_1 (with quantity $\bar{w}_{i_1t_1}$) and i_2 in period t_2 (with quantity $\bar{w}_{i_2t_2}$), where t_2 is the first period after t_1 in which $\bar{w}_{i_2t_2} > 0$. For customer i_1, the move considers the maximum portion of $\bar{w}_{i_1t_1}$ that can be reassigned to period t_2 without causing a shortage in period t_1 to be exchanged with the full amount $\bar{w}_{i_2t_2}$. (We show below that it is suboptimal to transfer less than the maximum portion of $\bar{w}_{i_1t_1}$). If customer i_1 was not scheduled for a delivery in period t_2, then he must be inserted into one of the θ routes. In general, a swap produces a change in holding costs and a change in transportations costs in periods t_1 and t_2. If the net effect is negative, then the swap is beneficial. Note once again that in the full model, the transportation costs are the sum of the routing costs c_{ij} and hence do not depend on the delivery quantities.

Proposition 4.2 Let $\bar{w}_{i_1t_1}$ be the planned delivery quantity to customer i_1 in period t_1 and let $\bar{I}^C_{i_1t_1}$ be the net inventory position after demand is satisfied; i.e., $\bar{I}^C_{i_1t_1} = \bar{I}^C_{i_1,t_1-1} + \bar{w}_{i_1t_1} - d_{i_1t_1}$. For any feasible swap between customers i_1 and i_2 in periods t_1 and t_2, the optimal swap amount is equal to $\min\left\{\bar{w}_{i_1t_1}, \bar{I}^C_{i_1t_1}\right\}$.

Proof. Consider any swap amount χ for customer i_1 in period t_1 such that $\chi < \min\left\{\bar{w}_{i_1t_1}, \bar{I}^C_{i_1t_1}\right\}$. We will show that there exists a swap amount χ^* such that $\chi < \chi^* \leq \min\left\{\bar{w}_{i_1t_1}, \bar{I}^C_{i_1t_1}\right\}$ that results in a better *move_value*.

The *move_value* is equal to the sum of the change in holding costs for customers i_1 and i_2, the change in transportation costs in periods t_1 and t_2, and the change in holding cost at the plant. For the case where the swap amount is χ, the *move_value* is $-h^C_{i_1}\chi + \varepsilon^1_{t_1} + \varepsilon^1_{t_2} + h^C_{i_2}\bar{w}_{i_2t_2} + \rho^1$, where $\varepsilon^1_{t_1}$ and $\varepsilon^1_{t_2}$ represent the change in transportation costs in period t_1 and t_2, respectively, and ρ^1 is the change in holding cost at the plant. For the latter, we have $\rho^1 = -h^P(\chi - \bar{w}_{i_2t_2})(\tau^1_{t_2} - \tau^1_{t_1})$, where $\tau^1_{t_1} \leq \tau^1_{t_2}$ are respectively the periods associated with t_1 and t_2 in which production must be adjusted to ensure that the swap is feasible. (Although the results are stated for a single pair of adjustment periods, $\tau^1_{t_1}$ and $\tau^1_{t_2}$, it is easy to show that they hold when production must be adjusted in more than these two periods).

Similarly, for the case where the swap amount is χ^*, the *move_value* is equal to $-h^C_{i_1}\chi^* + \varepsilon^2_{t_1} + \varepsilon^2_{t_2} + h^C_{i_2}\bar{w}_{i_2t_2} + \rho^2$. (The symbols have the same meaning as above but a superscript 2 is used instead of 1.) To compare transportation costs, the following two cases must now be considered.

Case 1: When $\chi^* < \bar{w}_{i_1t_1}$ (the situation when $\chi^* = \bar{I}^C_{i_1t_1}$) there is no difference in transportation costs, that is, $\varepsilon^1_{t_1} = \varepsilon^2_{t_1}$ and $\varepsilon^1_{t_2} = \varepsilon^2_{t_2}$, because the customers who are scheduled for a delivery in period t_1 are the same as those scheduled for a delivery in t_2.

Case 2: When $\chi^* = \bar{w}_{i_1t_1}$ it is not necessary to make a delivery to customer i_1 in period t_1 so $\varepsilon^2_{t_1} \leq \varepsilon^1_{t_1}$ in period t_1 and $\varepsilon^2_{t_2} = \varepsilon^1_{t_2}$ in period t_2.

Because $\chi^* > \chi$, more product is moved to a later period, which implies a greater reduction in the holding cost at plant: $\rho^2 < \rho^1$. Finally, by noting that the amount of

product moved to an earlier period, $\bar{w}_{i_2 t_2}$, is the same in both cases, we have

$$-h_{i_1}^C \chi + \varepsilon_{t_1}^1 + \varepsilon_{t_2}^1 + h_{i_2}^C \bar{w}_{i_2 t_2} + \rho^1 > -h_{i_1}^C \chi^* + \varepsilon_{t_1}^2 + \varepsilon_{t_2}^2 + h_{i_2}^C \bar{w}_{i_2 t_2} + \rho^2 . \quad \blacksquare$$

The second move is called a *transfer* and examines each customer i one at a time and tries to reassign the delivery quantity $\bar{w}_{it_1}$ scheduled for t_1 to the latest period, call it t_2, preceding t_1 in which a delivery is scheduled for at least one customer i; that is, $t_2 = \max\{t : t < t_1, \exists\, \bar{w}_{it} > 0 \text{ for some } i \in N\}$. In this case, we have the following result.

Proposition 4.3 For any customer i, when considering a transfer move between periods t_1 and t_2, it is suboptimal to reassign less than the full amount $\bar{w}_{it_1}$ to period t_2.

Proof. If only a portion of $\bar{w}_{it_1}$ were transferred, then the holding costs would increase in t_1 for customer i while the transportation costs would remain the same in both periods so there would be no benefit in considering intermediate cases. $\blacksquare$

Whether a swap or a transfer, only moves that result in feasible solutions are permitted so it is necessary to check for violations of the production constraints and the inventory bounds at the plant and the customer sites. Moreover, a VRP must be solved in periods t_1 and t_2 to see whether feasible routes can be found after the move, and if so, what their (optimal) costs are. The value of a move is determined in part by these costs, which must be calculated for each candidate. To begin, *Feasibility_Check_Algorithm* is called to determine whether the move violates any of the production, storage or vehicle capacity constraints. If not, then *Move_Value_Algorithm* is called to determine the net benefit (see Appendix A).

Complexity of neighborhood. A swap involves an exchange in delivery quantities between pairs of customers in two different periods $t_1 < t_2$, so the number of possible candidates in the worst case is $O(n^2)$ in each period. Calculating the portion of $\bar{w}_{i_1 t_1}$ that can be exchanged without causing a shortage in period t_1 can be done in $O(1)$. The

feasibility check is $O(n + \tau)$ and determining the move value by solving the VRP takes $O(n^3)$. For τ periods then, the amount of work associated with a swap is $O((n^3+n+\tau)\cdot n^2\tau)$ $= O(n^5\tau + n^2\tau^2)$. For transfer moves, deliveries to each of the n customers in period t are considered for reassignment to period $t - 1$. In the worst case, there are $O(n)$ possibilities in each period. Taking the feasibility check and the move value computations into account, the amount of work associated with a transfer for all τ periods is $O((n^3+n+\tau)\cdot n\tau)$ $= O(n^4\tau + n\tau^2)$. Thus, the neighborhood size is $O(n^5\tau + n^2\tau^2)$.

Example of moves. Figure 4.3 depicts a swap between customers 1 and 3 in periods 2 and 3, respectively. The periods are numbered on the far left and the customers on a specific route are represented by circles. The depot (customer 0) is at the start and end of each route, implying that a single vehicle only is required in each period. The parameter values for the five customers in the example are as follows. The holding cost for customer 1 is $h_1^C = 20$ while the holding costs for customers 2 to 5 are $h_2^C = h_3^C = h_4^C = h_5^C$ $= 10$. Initial inventories at all customer sites in period 2 are $I_{11}^C = I_{21}^C = I_{31}^C = I_{41}^C = I_{51}^C = 0$. It is also assumed that the storage capacity at the customer sites is unlimited and that the vehicle capacity $Q = 60$.

Beginning at the depot, route costs are calculated by summing the individual link costs, c_{ij}, between customers i and j on the route, where $c_{01} = 175$, $c_{04} = 10$, $c_{12} = 25$, $c_{02} = c_{20} = 50$, $c_{23} = 75$, $c_{03} = c_{30} = 25$, $c_{45} = 30$ and $c_{51} = 10$. The route cost in period 2, for example, is $c_{04} + c_{45} + c_{51} + c_{12} + c_{23} + c_{30} = 175$. Demand for customers 1 to 5 in period 2 is 8, 7, 7, 6 and 8, respectively, and in period 3 is 4, 4, 8, 6 and 8. These values are shown in the squares below the circles in Figure 4.3 only if a delivery is scheduled for the customer in the period of interest.

Before the swap, customer 1 is scheduled for deliveries of 8 and 4 items in periods 2 and 3, respectively, and customer 3 is scheduled for a delivery of 15 items in period 2. After the swap, the amount to be delivered to customer 1 is increased to 12 in period 2 and reduced to 0 in period 3. Also, the amount to be delivered to customer 3 is

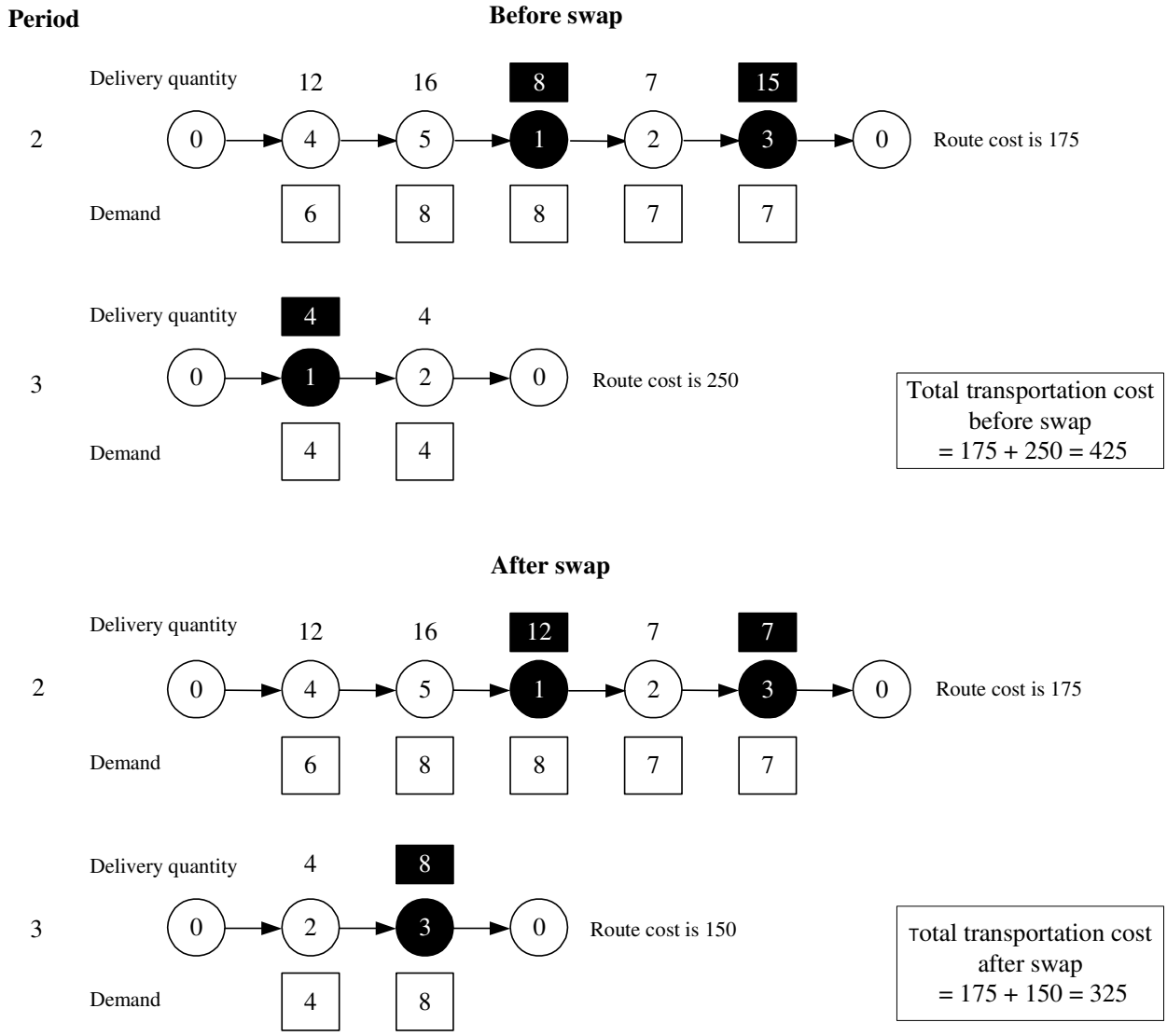

Figure 4.3. Example of an exchange or swap move between customers 1 and 3

decreased to 7 in period 2 in accordance with Proposition 2 and a new delivery of 8 units is scheduled in period 3. Thus, for customer 3, the exchange involves the maximum

number of items, 8, that can be moved in period 2 without causing a shortage (customer 3 has a demand of 7 in period 2 and currently, 15 items are scheduled for delivery). For customer 1, all 4 units that were to be delivered in period 3 are now scheduled for period 2. Note that before and after the swap the total number of items scheduled for delivery in either period do not exceed the vehicle capacity. Before the swap, a total of 58 and 12 items are scheduled to be delivered in periods 2 and 3, respectively, while after the swap, the delivery quantities are 54 and 16.

The results indicate that a big cost reduction is achieved by eliminating the link from the depot to customer 1 ($c_{01} = 175$) in period 3. This cannot be realized, though, by simply rescheduling the 4 items to be delivered to customer 1 in period 3, to period 2 because the vehicle capacity, 60, would be exceeded. A swap between periods is required. Before the swap the transportation cost in period 3 is $c_{01} + c_{12} + c_{20} = 250$ and after the swap it is $c_{02} + c_{23} + c_{30} = 150$, where the solution to the VRP for period 3 indicated that it was optimal to insert customer 3 after customer 2. In period 2, there is no change in route so the transportation cost remains the same, as do the holding cots. Calculating the difference between the before and after costs gives the *move_value*, which in this case is –100.

Using the same data and starting with the schedule in the bottom portion of Figure 4.3, Figure 4.4 gives an example of a single customer transfer move. Before the transfer, customer 2 is scheduled to receive deliveries of 7 and 4 items in periods 2 and 3, respectively, where now $t_1 = 3$ and $t_2 = 2$. The transfer eliminates the delivery in period 3 by moving all 4 items to period 2 in accordance with Proposition 3, and is feasible because the total load on the vehicle only goes up to 58, which is less than the capacity. The move increases the holding cost for customer 2 from 0 of 40, but the transportation cost in period 3 is reduced by 100, giving a *move_value* of –60. More specifically, after the transfer the net change in holding cost is $4 \times 10 = 40$ and is associated with customer 2, the total transportation cost in period 3 is $c_{0,3} + c_{3,0} = 50$, and the net change in transportation cost in period 2 is 0. As a result, the *move_value* is $50 + 40 - 150 = -60$.

4.4 TABU LIST AND ASPIRATION CRITERION

Tabu search transcends local optimality by forbidding certain moves on a temporary basis. In general, the process is implemented with a short-term memory structure that proscribes a subset of the moves in a neighborhood. The *tabu list* stores all forbidden moves. Its length, normally called the *tabu tenure*, indicates how many iterations a certain move is forbidden. The *tabu status* indicates whether a move on the tabu list is currently forbidden.

In our implementation, we use a reactive tabu list meaning that the tabu tenure is dynamic. We also specify an aspiration criterion that allows the tabu status of a move to be overridden. Specifically, each entry on the tabu list is a combination of a pair of customers and a pair of time periods represented by the 4-dimensional vector (customer i_1, customer i_2, period t_1, period t_2). Note that for a transfer move where only a single customer is involved, a default value of zero is used for customer i_2. The length of tabu list is kept constant as long as progress is being made, but it is increased when there is no improvement in some fixed number of iterations. Because a move associated with a customer in one period could create a series of moves in the following periods, thus affecting up to $\tau - 1$ periods, we set the tabu tenure proportional to τ. Its initial value is set to $\tau/2$ and then increased to τ when there is no improvement for 5 iterations. However, when a new incumbent is found, the tabu tenure is set back to $\tau/2$. The process is repeated until the tabu search stopping criteria are met.

Although some implementations allow infeasible moves as an additional means of overcoming local optimality, we do not. Preliminary testing showed that when infeasible moves were considered at each iteration, runtimes became excessive due to the increase in neighborhood size for even small instances with up to 20 customers.

After the current neighborhood is searched and a new incumbent is found, the move that led to the incumbent is added to the tabu list. The tabu status of a move can be overridden, though, when a certain aspiration condition is met. In our case, this condition is associated with a move on the tabu list that gives a better solution than the incumbent.

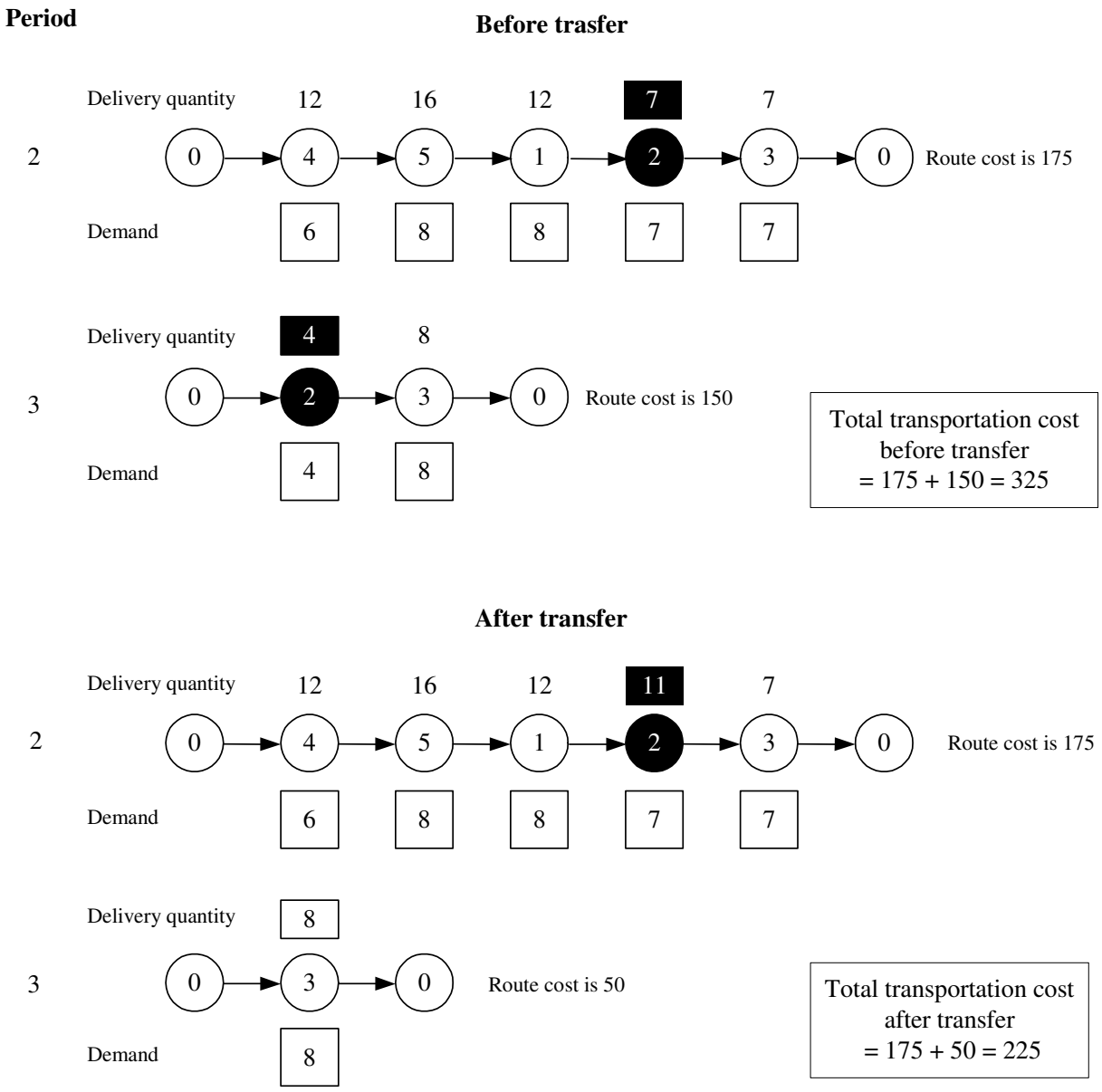

Figure 4.4. Example of a transfer move that assigns a delivery to an earlier period

4.5 SEARCH STRATEGY

Two important strategies for tabu search are intensification and diversification. Intensification focuses on creating solutions that have good attributes (with respect to routing, for example, solutions that include certain arcs). Diversification is the ability of the algorithm to expand the search by generating solutions that have attributes different from those encountered at previous iterations. These two strategies counterbalance and reinforce each other.

In our algorithm, intensification is implemented by using incentives and diversification by using penalties and short-term memory. The latter is provided by the tabu list. Incentives and penalties are a function of long-term memory and are represented by the $(n \times n \times \tau \times \tau)$-dimensional matrices $\boldsymbol{F^I}$ and $\boldsymbol{F^P}$, respectively. For a move that involves customer i_1 in period t_1 and customer i_2 in period t_2, an element $F^I_{i_1 i_2 t_1 t_2}$ of $\boldsymbol{F^I}$ represents the number of times the set $\{i_1, i_2, t_1, t_2\}$ has been involved in a move that improves the solution. Similarly, an element $F^P_{i_1 i_2 t_1 t_2}$ of $\boldsymbol{F^P}$ represents the number of times the set $\{i_1, i_2, t_1, t_2\}$ has been involved in a nonimproving move. Any move associated with these sets is either rewarded or penalized in accordance to the values of $F^I_{i_1 i_2 t_1 t_2}$ and $F^P_{i_1 i_2 t_1 t_2}$. With respect to the above example, if a candidate swap involves customer i_1 in period t_1 and customer i_2 in period t_2, then the actual value of the move is *move_value* – $\rho^I F^I_{i_1 i_2 t_1 t_2} + \rho^P F^P_{i_1 i_2 t_1 t_2}$ rather than m*ove_value* alone, where ρ^I and ρ^P represent incentive and penalty multipliers (see next section for more detail).

Implementing exhaustive search for the neighborhood defined in Section 4.3 is possible for small ($n^2\tau \leq 50$) and intermediate size problems ($50 < n^2\tau \leq 500$). For large instances ($n^2\tau > 500$) it is not practical to examine all candidate moves. To reduce the computational effort, we randomly select a subset of moves and then adaptively decide according to the progress of the algorithm, which of two rules to follow. The first rule places a 4 to 1 emphasis on transfers over swaps and is used when the most recent tabu iteration resulted in an overall cost reduction. The second rule reverses the emphasis and is used when the most recent tabu iteration did not improve upon the incumbent. Let M_S

be the set of feasible swaps and let M_T be the set of feasible transfers at some iteration of the algorithm. The following steps are performed.

1. Based on the solution from the previous tabu iteration (if this is the first iteration, use the initial solution), create the two sets M_S and M_T.
2. For each member of M_S and M_T, draw a number from the [0,1] uniform distribution and call it ξ.

3a. Rule 1: For each member of M_T, if $\xi \in (0.5, 0.9]$, place it on the list of candidate moves; otherwise discard it. For each member of M_S, if $\xi \in (0.9, 1]$, place it on the list of candidate moves; otherwise discard it.

3b. Rule 2: For each member of M_S, if $\xi \in (0.5, 0.9]$, place it on the list of candidate moves; otherwise discard it. For each member of M_T, if $\xi \in (0.9, 1]$, place it on the list of candidate moves; otherwise discard it.

At this point, all moves on the candidate list are processed and the one with the best *move_value* is selected. The relevant production and inventory levels are updated, the stopping criteria are checked, and if not met, the next tabu iteration is performed. The rationale for the two rules is as follows. In general, the solution from the allocation model results in lower vehicle utilization than the true optimum because the transportation costs in (4-1a) are overestimated. Recall that they are set equal to the roundtrip cost between the depot and the customer sites. As a consequence, the allocation model favors holding items in inventory over more frequent deliveries so in some periods, only a fraction of a vehicle's capacity may be used. Transfer moves are aimed at increasing the load on a vehicle by trying to consolidate deliveries that are scheduled for two periods down to one. Because transfers typically result in larger improvements than swaps, they are more powerful and hence used whenever the most recent tabu iteration yields an overall cost reduction. When this is not the case, the delivery schedule must be changed if any improvement is to be realized within our neighborhood definition. Rule 2 places the emphasis on swaps and serves this purpose. Swaps investigate the benefits of exchanging customers between periods but do not

necessarily lead to the elimination of a customer on a route. As such, any cost reductions that result are likely due to better delivery schedules than to higher vehicle utilization.

4.6 ALGORITHM DESCRIPTION

Figure 4.5 summarizes the major steps of the tabu search algorithm for large instances. In the first step, an initial feasible solution is created with the procedure described in Section 4.1. The result is saved as both the *current solution* and the *best solution* found. Then the algorithm iterates until either a prespecified *max_ iterations* is reached or there is no improvement in *max_no_improve* iterations. Of all the admissible candidates, the move selected at each iteration is the one that has the smallest value of *move_value* + *move_penalty* + *move_incentive*. A move is considered admissible if it is not on the tabu list or its tabu status has been overridden by the aspiration criterion. For the current iteration, once the best move is found it is executed and the tabu data structures are updated.

The parameters and data structures used in the algorithm are as follows.

- *num_iteration*: current iteration number
- *max_iterations*: maximal number of iterations allowed
- *curr_soln*: current solution after executing the best move
- *curr_cost*: cost of current solution
- *best_soln*: incumbent solution
- *best_cost*: cost of incumbent solution
- *move_value*: difference in the cost before and after the swap move
- *best_move_value*: lowest *move_value* among all candidates
- *freq_matrix*: long-term memory function that stores the frequency of each possible move
- *move_penalty*: additional penalty imposed by long-term memory *freq_matrix* $\boldsymbol{F}^{\boldsymbol{P}}$; when a swap involves customer i in period k and customer j in period l, *move_penalty* =

$\rho^P F^P_{ijkl}$, ρ^P is set to 5% of the objective function value of the initial feasible solution divided by *max_iterations*. This value gives an approximation of the average amount of penalty per iteration.

- *move_incentive*: additional incentive imposed by long-term memory *freq_matrix* $\boldsymbol{F^I}$; when a swap involves customer *i* in period *k* and customer *j* in period *l*, *move_incentive* $= -\rho^I F^I_{ijkl}$, where ρ^I is set to ρ^P because we penalize a move that results in a non-improved solution the same amount as the incentive of the same move that results in an improved solution.
- *best_move_penalty*: *move_penalty* for the best move of all candidates
- *best_move_incentive*: *move_incentive* for the best move of all candidates
- *no_improve*: number of consecutive iterations during which no better solutions are found
- *max_no_improve*: maximum number of consecutive no improvement iterations allowed
- *tabu_list*: short-term memory function that stores a list of moves that are forbidden
- *tabu_size*: total number of iterations for which a move is held on the *tabu_list*
- *candidate_rule*: candidates that are considered during each iteration in accordance with rules 1 and 2
- *candidate_move*: solution that results when the moves associated with *candidate_rule* are applied
- *Admissible_move*: candidate move that either satisfies the aspiration criterion or is not on the *tabu_list*

```
1.  Generate initial feasible solution and save it as curr_sol and best_soln.
2.  Calculate the cost of curr_sol and set best_cost = curr_cost.
3.  Initialize tabu_list and freq_matrix, set no_improve = 0 and set candidate_rule = 1.
4.  Do{
        best_move_value = ∞
        for (all candidate_move)
        {
            if(candidate_move is not on tabu_list or if it satisfies aspiration criterion )
            {
                if(Feasibility_Check_Algorithm return <true>)
                {
                    call Move_Value_Algorithm, store result in move_value
                    if(move_value + move_penalty + move_incentive <  best_value +
                        best_move_penalty + best_move_incentive)
                    {
                        best_move_value = move_value
                        best_move_penalty = move_penalty
                        best_move_incentive = move_incentive
                        best_move = current_move
                    }
                }
            }
        }
        execute best_move
        curr_cost = curr_cost + best_move_value
        update tabu_list and freq_matrix and adjust production levels
        if(curr_cost < best_cost)
        {
            best_cost = curr_cost
            best_soln = curr_soln
            no_improve = 0
            set candidate_rule = 1
         }else{
                no_improve = no_improve + 1
                set candidate_rule = 2
         }
    }While(num_iteration < max_iterations or no_improve < max_no_improve)
```

Figure 4.5. Tabu search algorithm

When evaluating each candidate move, it may be possible to achieve a further cost reduction by adjusting the production (and inventory) levels which are not necessarily optimal for the updated delivery schedule. Given the updated values of w_{it}, Proposition 4.1 allows us to solve a linear program to find the optimal production and corresponding inventory levels; however, doing so at each iteration is too costly because of the large number of possible moves, so in the implementation exact solutions are computed once every five iterations. At all other times, we attempt to improve the solution by calling the heuristic *Production_Level_Adjustment_Algorithm*. See Appendix A for the details.

Chapter 5

Lower Bound Computation

To gauge the quality of the solutions provided by tabu search, lower bounds on the true optimum are needed. The simplest way to obtain a lower bound is to solve the LP relaxation of the full model. Initial testing on small instances showed gaps of roughly 30% to 260% -- not a useful measure. As an alternative, we developed a procedure based on the allocation model that gives much better results.

To obtain a lower bound on the true optimum, and hence on the feasible solutions provided by tabu search, a valid relaxation of the full model must be solved. To formulate such a relaxation, we begin with (4-1a) – (4-1j) and make several modifications. First, the cost coefficients f_{it}^C for each customer i are set to the shortest distance to either the depot or any of the other customers, that is, and $f_{it}^C = \min\{c_{ij} : j \in N_0 \setminus (i)\}$. Next, the cost coefficients associated with w_{it} are set as follows: $e_{it}^C = c_0/Q$, where $c_0 = \min\{c_{0j} : j \in N\}$. Finally, the right-hand side of (4-1h) is increased to $Q\theta$ in each period t and the corresponding allocation model solved to get ϕ_{LB}.

Proposition 5.1 The solution to the modified allocation model (lower bounding model) provides a valid lower bound on the true optimum to the PIDRP; that is, $\phi_{\text{LB}} \le \phi_{\text{PIDRP}}$.

Proof. First note that any solution that is feasible to the full model is feasible to the modified allocation model. This follows because (4-1h) with its right-hand side set to $Q\theta$ is a valid relaxation of the routing constraints in the full model. We now show that the modified costs $\sum_{it} f_{it}^C z_{it}^C + \sum_{it} e_{it}^C w_{it}$ underestimate the true costs $\sum_{ijt} c_{ij} x_{ijt}$, which in any feasible solution contain one arc cost for each customer that receives a delivery in period t and one arc cost for each vehicle used in period t. By design, the first term

underestimates the individual customer arc costs and the second underestimates the vehicle arc costs from the depot. With respect to the vehicle arc costs, note that the number of vehicles required to deliver $\Sigma_{it} w_{it}$ units in period t is at least $\lceil \Sigma_{it} w_{it} / Q \rceil$. Removing the integer requirement and multiplying by c_0 gives the desired result. ■

A slightly improved lower bound can be obtained when the holding costs are small and production setup costs are large with respect to the vehicle cost, which is the case here.

5.1 IMPROVED LOWER BOUND WHEN HOLDING COSTS ARE SMALL AND PRODUCTION SETUP COSTS ARE LARGE

Lemma 5.1 Assume that the minimum production setup cost $f^P \equiv \min\{f_t : t = 0,1,\ldots,\tau-1\}$ is much larger than the cost c_0 of using a vehicle in any period, and that the customer holding costs are negligibly small, i.e., $h_i^C \approx 0$, $\forall\, i \in N$. Let $R(t) = \left\lceil \sum_i w_{it} / Q \right\rceil - \sum_i w_{it} / Q$ and $r(t) = \sum_i w_{it} / Q - \left\lfloor \sum_i w_{it} / Q \right\rfloor$ be the fraction of a vehicle corresponding to the difference between the rounded up and rounded down number of vehicles in a solution in period t, respectively. Under the stated conditions, if

$$c_0 r(t) + (t' - t) h^P r(t) Q - (t'_P - t_P) h^P r(t) Q - c_0 - c_0 R(t') \geq 0, \ \forall\ t' > t \tag{5-1}$$

$$c_0 r(t) - (t - t') h^P r(t) Q + (t_P - t'_P) h^P r(t) Q - c_0 - c_0 R(t') \geq 0, \forall\ t' < t \tag{5-2}$$

then the lower bound ϕ_{LB} can be increased by rounding up the number of vehicles used in period t and multiplying the corresponding fraction $R(t)$ by c_0. Summing over all periods gives the following adjusted lower bound.

$$\phi_{\text{LB}} \leftarrow \phi_{\text{LB}} + \sum_{t=1}^{\tau} c_0 \left(\left\lceil \sum_i w_{it} / Q \right\rceil - \sum_i w_{it} / Q \right) \tag{5-3}$$

Proof. The assumption that $f^P >> c_0$ implies that it is never advantage to set up for production in order to avoid using an additional vehicle in any period. Therefore, the result follows if this is the only feasible option for eliminating a vehicle in some period t.

Rounding up number of vehicles used in each period will not change the optimality of the solution if the increase in cost, $c_0 \cdot R(t)$, in period t does not exceed the cost of rescheduling the delivery quantity $Q \cdot r(t)$ to another period t'.

Case 1: $t' > t$

Given ϕ_{LB}, the objective function value when rounding up the number of vehicles used without rescheduling in both periods t and t' is

$$\text{Cost}^1 = c_0 R(t) + c_0 R(t') + \phi_{\text{LB}} \tag{5-4}$$

Now, let t_P and t'_P be the periods where production of $Q \cdot r(t)$ takes place before and after rescheduling, respectively. The objective function value after rescheduling is bounded below by

$\text{Cost}^2 =$

$$\varphi_{\text{LB}} - c_0 r(t) + (t' - t) h^P r(t) Q - (t'_P - t_P) h^P r(t) Q - (t' - t) \max\{h_i^C, i \in N\} r(t) Q +$$
$$c_0 \left(\left(\sum_i w_{it'} + r(t) Q \right) / Q - \sum_i w_{it'} / Q \right) \tag{5-5}$$

Thus, if $\text{Cost}^2 \geq \text{Cost}^1$ for all $t' > t$, then rounding up the number of vehicles used in period t does not change the optimality of the solution.

Case 2: $t' < t$

Using the same logic, the objective function value when rounding up the number of vehicles used after rescheduling is bounded below by

$\text{Cost}^3 =$

$$\phi_{\text{LB}} - c_0 r(t) - (t - t') h^P r(t) Q + (t_P - t'_P) h^P r(t) Q + (t - t') \min\{h_i^C, i \in N\} r(t) Q +$$
$$c_0 \left(\left(\sum_i w_{it'} + r(t) Q \right) / Q - \sum_i w_{it'} / Q \right) \tag{5-6}$$

The implication of (5-4) and (5-6) is that rounding up the number of vehicles used in period t does not change the optimality of the solution if $\text{Cost}^3 \geq \text{Cost}^1$ for all $t' < t$.

Now, when $h_i^C \approx 0$, $\forall\, i \in N$, (5-4), (5-5) and (5-6) give

Case 1: $t' > t$

$$c_0 r(t) + (t' - t) h^P r(t) Q - (t'_P - t_P) h^P r(t) Q - c_0 - c_0 R(t') \geq 0$$

Case 2: $t' < t$

$$c_0 r(t) - (t - t') h^P r(t) Q + (t_P - t'_P) h^P r(t) Q - c_0 - c_0 R(t') \geq 0$$

■

5.2 IMPROVED LOWER BOUND BY CONSIDERING THE MINIMAL NUMBER OF VEHICLES RETURNED TO DEPOT

A second and third improvement in the lower bound can be obtained by taking into account the fact that exactly $\lceil \Sigma_{it} w_{it} / Q \rceil$ vehicles must return to the depot in any solution to the full model. A fourth improvement can be obtained by recognizing that some minimum number of vehicles must be used in each period in order to meet demand.

Lemma 5.2 Let $y_t = \lceil \Sigma_{it} w_{it} / Q \rceil$ be the minimum number of vehicles used in period t and let $c_i = \min\{c_{ij} : j \in N\}$ be the least cost transition for customer i to another customer. Also, let $N^*(t) = \{i : z^C_{it} = 1, f^C_{it} = c_{i0}\}$ be the set of customers who receive a delivery in period t and whose corresponding cost is c_{i0}, and let $\Delta c_i = c_i - c_{i0}$ be the opportunity cost of not using the minimum cost arc in a solution. Now, order the $n^*(t) = |N^*(t)|$ customers such that $\Delta c_{i_1} \leq \Delta c_{i_2} \leq \cdots \leq \Delta c_{i_{n^*(t)}}$ and let $l(t) = n^*(t) - y_t$ be the excess number of customers whose delivery cost is c_{i0}. Assume that the minimum production setup cost $f^P = \min\{f_t : t = 0,1,\ldots,\tau-1\}$ is much larger than the cost c_0 of using a vehicle in any period. If

(i) $\Delta c_i \leq (t' - t) w_{it} h^P - (t'_p - t_P) w_{it} h^P - (t' - t) w_{it} h^C_i - f^C_{it}$, $\forall\ t' > t$, and

(ii) $\Delta c_i \leq -(t - t') w_{it} h^P + (t_P - t'_p) w_{it} h^P + (t - t') w_{it} h^C_i - f^C_{it}$, $\forall\ t' < t$,

then the lower bound ϕ_{LB} can be adjusted as follows

$$\phi_{\text{LB}} \leftarrow \phi_{\text{LB}} + \sum_{t=1}^{\tau} \sum_{k=1}^{l(t)} \Delta c_{i_k} \tag{5-7}$$

Proof. Increasing the cost of making delivery to customer $i \in N^*(t)$ by Δc_i in period t will not change the solution to the modified allocation model if the incurred cost Δc_i is less than the cost of rescheduling delivery to customer i in period t to another period t'.

Case 1: $t' > t$

Assuming that rescheduling is feasible, moving a delivery to a later period decreases customer i's holding cost by $(t' - t)w_{it}h_i^C$. In contrast, the holding cost at the plant changes by $-(t - t_P)w_{it}h^P + (t' - t'_p)w_{it}h^P$, where t_P and t'_p are the periods in which w_{it} is produced before and after rescheduling, respectively (it is likely that $t'_p \geq t_P$ but not guaranteed). When the delivery quantity w_{it} is rescheduled for period t', these changes imply that the lower bound can be adjusted without affecting the solution if $\Delta c_i \leq (t' - t)w_{it}h^P - (t'_p - t_P)w_{it}h^P - (t' - t)w_{it}h_i^C$. In the worse case, a deliveries will have to be scheduled for customer i in both periods t' and t so the condition becomes $\Delta c_i \leq (t' - t)w_{it}h^P - (t'_p - t_P)w_{it}h^P - (t' - t)w_{it}h_i^C - f_{it}^C$.

Case 2: $t' < t$

Assuming once again that rescheduling is feasible, moving a delivery to an earlier period increases customer i's holding cost by $(t - t')w_{it}h_i^C$, while the holding cost at the plant changes by $-(t - t')w_{it}h^P + (t_P - t'_p)w_{it}h^P$. This implies that the lower bound can be adjusted without affecting the solution if $\Delta c_i \leq -(t - t')w_{it}h^P + (t_P - t'_p)w_{it}h^P + (t - t')w_{it}h_i^C$. When a delivery is scheduled for customer i in both periods t' and t, the condition becomes $\Delta c_i \leq -(t - t')w_{it}h^P + (t_P - t'_p)w_{it}h^P + (t - t')w_{it}h_i^C - f_{it}^C$.

The statement of the lemma follows from these developments. ■

When the holding cost at customer i is negligibly small, that is, $h_i^C \approx 0$ for all $i \in N$ (the case for our data), then conditions (i) and (ii) in Lemma 5.2 become

(i) $\Delta c_i \le (t' - t) w_{it} h^P - (t'_p - t_P) w_{it} h^P - f_{it}^C$, $\forall\ t' > t$

(ii) $\Delta c_i \le -(t - t') w_{it} h^P + (t_P - t'_p) w_{it} h^P - f_{it}^C\ \ \forall\ t' < t$

Lemma 5.3 Expanding on the notation introduced in Lemma 5.2, let $L^*(t) = \{i : z_{it}^C = 1, f_{it}^C \ne c_{i0}\}$ be the set of customers who receive a delivery in period t and whose corresponding cost is not c_{i0}. For $i \in L^*(t)$, let $\Delta c_i = c_{i0} - c_i$ be the increase in cost that would result if customer i was assigned the arc connected to the depot in a solution instead of c_i. Assume that $y_t > n^*(t)$ and let $l(t) = y_t - n^*(t)$ be the number of customers whose delivery cost should be set to c_{i0}. Now identify the $l(t)$ smallest values of Δc_i for $i \in L^*(t)$; that is, $\Delta c_{i_1} \le \Delta c_{i_2} \le \cdots \le \Delta c_{i_{l(t)}}$, where $i_1, ..., i_{l(t)} \in L^*(t)$. Assuming that the minimum production setup cost $f^P = \min\{f_t : t = 0,1,\ldots,\tau-1\}$ is much larger than the cost c_0 of using a vehicle in any period, if

(i) $\Delta c_i \le (t' - t) w_{it} h^P - (t'_p - t_P) w_{it} h^P - (t' - t) w_{it} h_i^C - f_{it}^C$, $\forall t' > t$, and

(ii) $\Delta c_i \le -(t - t') w_{it} h^P + (t_P - t'_p) w_{it} h^P + (t - t') w_{it} h_i^C - f_{it}^C$, $\forall t' < t$

then the lower bound ϕ_{LB} can be adjusted as follows

$$\phi_{\text{LB}} \leftarrow \phi_{\text{LB}} + \sum_{t=1}^{\tau} \sum_{k=1}^{l(t)} \Delta c_{i_k} \tag{5-8}$$

Proof. Similar to that of Lemma 5.2 ■

Lemma 5.4 Define $v(t)$ as the minimum number of vehicles needed in period t. Let $i^* = \arg\min\{c_{i0},\ i \in N\}$ and let $\Delta c_{it} = c_{0i} - c_0$ be the incremental cost for customer $i \in N \backslash \{i^*\}$ in period t for not using the minimum cost arc (with cost c_0) from the depot in a solution. For period t, identify the $v(t) - 1$ smallest values of Δc_{it} and the corresponding customers. The following adjusted lower bound is valid for the modified allocation model.

$$\phi_{\text{LB}} \leftarrow \phi_{\text{LB}} + \sum_{t=1}^{\tau} \sum_{k=1}^{v(t)-1} \Delta c_{i_k t} \tag{5-9}$$

Proof. The number of outbound arcs from the depot to customers in each period must be at least $v(t)$ in any feasible solution to the full model. Because each customer can be visited only once in each period, this implies that there must be outbound arcs from the depot cannot be the same for all vehicles. By selecting the $v(t) - 1$ smallest incremental costs associated with the arcs leaving the depot in period t, we increase ϕ_{LB} by $\sum_{k=1}^{v(t)-1} c_{i_k t}$ while simultaneously reducing it by $c_0(v(t) - 1)$. Summing over the planning horizon gives a valid lower bound on the total vehicle costs because we are only adjusting the cost for one less than the minimum number of vehicles required in each period. ■

The bound improvement in (5-9) can be calculated in a preprocessing step. The other improvements given in (5-3), (5-7) and (5-8) can be obtained directly by solving the allocation model after several additional modifications are made. For (5-3), it would be necessary to replace $\Sigma_{it} w_{it} / Q$ in the objective function with a nonnegative integer variable y_t ($t \in T$), and set the right-hand side of (3-1h) to Qy_t. For (5-7), it would be necessary to introduce $n \times \tau$ binary variables ξ_{it} ($i \in N$, $t \in T$), to identify whether cost coefficient c_{i0} or c_i is to be used for customer i in period t, and $n \times \tau$ constraints of the form $x_i + \xi_{it} \leq 1$ to ensure that at most one coefficient is selected. A similar modification would be required for (5-8). Testing has shown that these additions can measurably extend computation times so we have relied on Lemmas 5.1 – 5.4 to improve the initial bound.

Chapter 6

Computational Result – Tabu Search

All computations were performed on a 2.53 GHz processor with 512 MB of RAM. The optimization models and the tabu search code were implemented in Java Netbean 4.1 and linked to the CPLEX 8.1 libraries. CPU times were obtained through both CPLEX and the time function in Java.

For testing purposes, we used the three data sets provided by Boudia et al. (2007) containing 30 instances of 50, 100 and 200 customer problems, all with a 20-period planning horizon and holding costs $h^P = 1$, $h_i^C = 0$ for all $i \in N$. These instances were randomly generated on a 100×100 Euclidean grid. For each customer i, demand was uniformly distributed between 0 and the storage capacity $I_{\max .i}^C$, and for each period t was positive with a probability that varied from 0.5 to 1. The value of $I_{\max .i}^C$ depended on i and the number of customers in the specific data set. The vehicle capacities were $Q_{50} = 8000$, $Q_{100} = 8000$, $Q_{200} = 12000$, and the number of vehicles were $\theta_{50} = 5$, $\theta_{100} = 9$, $\theta_{200} = 13$.

The first set of experiments was designed to gauge CPLEX's performance on the full problem and did not make use of the Boudia et al. data sets.

6.1 PRELIMINARY TESTING

To determine the limits of CPLEX to solve the PIDRP directly, we randomly generated 10 instances on a 100×100 grid. The first step was to fix the number of customers $n \in \{5, 10, 15, 20\}$ and the number of time periods $\tau \in \{2, 4, 6, 8\}$ by selecting a range of values from these sets. The remaining parameter values were chosen as follows.

- Each customer location was represented by (x,y)-coordinates placed randomly on a 100×100 grid. The Euclidean norm was used to measure the distance between a pair of customers and the transportation cost was assumed to be proportional to that distance. In our case, a proportionality constant of 1 was used.
- Setup and holding costs at the plant, and demand and holding costs at the customer sites were generated from a uniform distribution with given upper and lower bounds:
    - Setup cost at the plant on day t, f_t: upper bound = 1000 and lower bound = 500 (values of f_t were generated independently for each day $t \in T$)
    - Holding cost at the plant, h^P: upper bound = 200 and lower bound = 100
    - Demand on day t, d_{it}: upper bound = 10 and lower bound = 0 (these values imply that for each day $t \in T$, 10% of customers have a chance of having zero demand)
    - Holding cost at customer sites, h_i^C: upper bound = 100 and lower bound = 50 for all $i \in N$
- Initial inventories at the plant as well as at the customer sites were set to zero; $I_\tau^P = I_{i\tau}^C = 0, \forall\, i \in N$
- The number of vehicles θ was fixed at 5 and the vehicle capacity Q was determined by calculating the total demand on each day, identifying the largest values, and multiply it by three; $Q = 3 \times \max\left\{\sum_{i \in N} d_{it} : t = 1,..., \tau\right\}$.
- No limit was imposed on the production capacity or the maximum inventory allowed at the plant; $C = I_{\max}^P = \infty$
- Maximum inventory allowed at a customer site was about three days of demand for that customer

The results are summarized in Table 6.1. Columns 2 and 3 are the number of time periods τ and number of customers n, respectively. Columns 4 and 5 represent the size of

the test problems as measured by the number of constraints and variables. Columns 6 and 7 are the LP solutions (ϕ_{LP}) at the root node and the runtimes (t_{LP}). Columns 8 and 9 report the solutions from CPLEX (ϕ_{cplex}) and the corresponding runtimes (t_{cplex}) in seconds. The computations were halted if the optimal solution was not found within 2 hours or the computer's memory limit was exceeded. The MipGap termination parameter in CPLEX was set to 0 in all runs. The last column reports the gap between the solution found by CPLEX and the LP solution at the root node. This value is give by gap = $((\phi_{cplex} - \phi_{LP}) / \phi_{LP})100\%$.

From Table 6.1 we can see that when either the number of customers or number of periods is increased, the runtime increases dramatically in almost all cases. This was to be expected because the PIDRP requires the solution of a CVRP in each period. When the number of customers is increased, the number of binary variables and constraints in each CVRP increases by $O(n^2)$ in the worst case, implying a significant increase in both the number of binary variables and constraints in the PIDRP model. For example, consider problems 5 to 7 where there are 4 time periods. When the number of customers increases from 5 to 10, the number of binary variables increases by a factor of 3.6 and the number of constraints by a factor of 3; when the number of customers increases from 5 to 15 the number of binary variables increases by a factor of 7.7 and the number of constraints by a factor of 6. Collectively, this shows that the model grows at a rate slightly less that $O(n^2)$.

In all instances, the LP solution was obtained in less than a second but runtimes for CPLEX ranged from 4 to > 7,200 sec. Notice that the problems that could not be solved by CPLEX are the ones that have $n\tau \geq 40$. The last column in Table 6.1 shows the large gap between the LP solution and the best solution found by CPLEX within the 2-hour limit. As discussed by Laporte (1992) and others, poor performance like this is due primarily to the weak relaxation associated with the load constraints (1j), which are rarely if ever binding in the LP solution. In the case of problems 7 and 10, CPLEX terminated prematurely at the point where its resource requirements exceeded the memory of the computer.

Table 6.1. Results from solving the PIDRP directly with CPLEX

Problem no.	τ	n	Number of constraints	Number of variables (total, binary)	(ϕ_{LP}) LP solution	(t_{LP}) LP time (sec)	(ϕ_{cplex}) CPLEX solution	(t_{cplex}) Runtime (sec)	Gap at root node (%)
1	2	5	99	(111, 63)	1,416.26	< 1	2,124	4	49.97
2	2	10	289	(311, 223)	1,518.75	< 1	1,983	16	30.57
3	2	15	579	(611, 483)	1,458.04	< 1	1,931	152	32.44
4	2	20	969	(1011, 843)	1,537.11	< 1	2,197	> 7,200	42.93
5	4	5	195	(209, 125)	1,419.29	< 1	3,014	196	112.36
6	4	10	575	(599, 445)	1,964.82	< 1	4,500	> 7,200	129.03
7	4	15	1155	(1189, 965)	2,297.93	< 1	4,574	> 6,435	97.87
8	6	5	291	(307, 187)	1,816.51	< 1	5,305	494	192.04
9	6	10	861	(887, 667)	2,217.78	< 1	5,857	> 7,200	164.09
10	8	5	387	(405, 249)	1,999.25	< 1	7,346	> 5,030	267.44

To test the performance of the VRP subroutine (Carlton and Barnes 1996) we created eight single-period problem instances in which the numbers of customers ranged from 5 to 150. The same generating procedure used for the PIDRP instances were used here but the inputs were limited to customer locations, customer demand, vehicle capacity, and number of vehicles.

The results are reported in Table 6.2 The first column lists the problem number and second column gives the number of customers associated with each instance. Columns 3 and 4 are the solutions from the VRP subroutine and the corresponding runtimes (ϕ_{VRP} and t_{VRP}). For problem nos. 11 - 16, the maximum number of tabu iterations was set to 100; for the remaining instances it was set to 200. Column 5 gives the best solution found by CPLEX (ϕ_{cplex}) within two hours and column 6 gives the corresponding runtimes (t_{Cplex}). The last column reports the percentage deviation of the VRP subroutine solutions from the solutions provided by CPLEX.

From the results in Table 6.2 we can see that the VRP subroutine gives high quality solutions within a much small amount of time compared to CPLEX, and is

especially effective for the larger instances with 30 or more customers. For problem no. 17, for example, $t_{VRP} = 4.6$ whereas t_{cplex} is > 7,200. For the small size problems ($n = 5$, 10, 20), the VRP subroutine provided the same solution as CPLEX within about the same runtime. In particular, the gap between ϕ_{VRP} and ϕ_{cplex} was less than or equal to zero in all but one case. Also, the VRP subroutine runtimes were substantially less than those of CPLEX, with the difference increasing dramatically as the number of customers increased.

Table 6.2. Performance of the capacitated vehicle routing subroutine

Problem no.	Number of customers	(ϕ_{VRP}) Solution from VRP subroutine	(t_{VRP}) Runtime (sec)	(ϕ_{cplex}) CPLEX solution	(t_{cplex}) Runtime (sec)	[(ϕ_{VRP}–ϕ_{cplex}) /ϕ_{cplex}]100%
11	5	133	< 1	133	< 1	0
12	10	271	< 1	271	< 1	0
13	20	321	< 1	321	236.19	0
14	30	450	< 1	443	336.64	1.6
15	40	536	1.0	598	> 7,200	–10.4
16	50	589	1.7	956	> 7,200	–37
17	100	1,012	4.6	1,573	> 7,200	–36
18	150	1,341	14.0	2,009	> 7,200	–33

Table 6.3 compares the phase 1 solutions obtained from the allocation model (4-1) with route optimization in each period with the solutions from CPLEX. Problems 1 through 10 were used in the testing. Columns 2 and 3 give the phase 1 solutions (ϕ_{Ph1}) and the corresponding runtimes (t_{Ph1}) in seconds, respectively. Columns 4 and 5 report the best solutions found by CPLEX (ϕ_{cplex}) along with the runtimes (t_{cplex}). The next column gives the deviation of the phase 1 solutions from the CPLEX solutions.

From Table 6.3 we can see that phase 1 reliably provides high quality solutions in a negligible amount of time compared to CPLEX. For problems 2 and 5, the optimal solution was found by the phase 1 procedure and in the other cases it was at most 6% off;

however, on 4 of the 10 instances, it provided better results than CPLEX. Also, the phase 1 runtimes were no more than a few seconds, not increasing much with either n or τ. For example, problems 1 and 10 both contain 5 customers with respectively 2 and 8 periods each. While τ increases by a factor of 4, the runtime went up by only 86%. For CPLEX, the increase was > 1000%. In addition, CPLEX either ran out of time or memory in half the instances.

Table 6.3. Comparison of solutions from phase 2 and CPLEX

Problem no.	(ϕ_{Ph1}) Phase 1 solution	(t_{Ph1}) Runtime (sec)	(ϕ_{cplex}) CPLEX solution	(t_{cplex}) Runtime (sec)	$[(\phi_{Ph1}-\phi_{cplex}) / \phi_{cplex}]100\%$	(ϕ_{Ph2}) Phase 2 solution	(t_{Ph2}) Runtime (sec)	$[(\phi_{Ph2}-\phi_{Ph1}) / \phi_{Ph1}]100\%$
1	2,189	0.78	2,124	4	3	2,124	0.24	–3
2	1,983	0.44	1,983	16	0	1,983	0	0
3	2,056	0.46	1,931	152	6	1,960	1.79	–5
4	2,089	0.53	2,197	> 7,200	–5	2,050	4.12	–2
5	3,014	1.98	3,014	196	0	3,014	0	0
6	4,165	0.45	4,500	> 7,200	–7	3,995	2.49	–4
7	4,493	1.29	4,574	> 6,435	–1	4,362	6.7	–3
8	5,337	3.39	5,305	494	0.6	5,305	0.62	–1
9	5,816	0.79	5,857	> 7,200	–0.7	5,647	3.36	–3
10	7,518	1.45	7,346	> 5,030	2.3	7,328	0.97	–3

Columns 7 and 8 of Table 6.3 give the phase 2 tabu search solutions (ϕ_{Ph2}) and runtimes (t_{Ph2}), respectively. The last column shows the percentage difference between the two. The improvement ranged from 0 for problems 2 and 5 that were solved optimally in phase 1, up to 5% for problem no. 3. The average gap between ϕ_{Ph2} and ϕ_{Ph1} for those instances that could be improved was 3%. For problems 1 and 8, the phase 2 solutions were provably optimal. Notice also that like t_{Ph1}, t_{Ph2} did not increase substantially with the problem size. This result can be attributed to the fact that the phase 1 solutions are often near optimal. In such cases, few candidate moves exist that can improve the phase 2 solution so the tabu search algorithm terminates quickly.

6.2 SMALL INSTANCES

Table 6.4 summarizes the results for the 50-customer, 20-time period instances. Columns 2 and 3 give the best GRASP solutions (ϕ_{grasp}) from Boudia et al. and the corresponding runtimes (t_{grasp}) for 500 iterations. Their computations were performed on a 2.8 GHz computer with 512 MB of RAM running Windows XP. As a word of caution, it is always problematic to make comparisons across different platforms and different algorithms. There is also some arbitrariness in establishing the termination criteria for metaheuristics like GRASP or tabu search because it is rarely evident when additional iterations will not lead to improvement. Therefore, all runtimes and other performance measures presented below should be interpreted with this in mind.

Columns 4, 5 and 6 are phase 1 solutions (ϕ_{Ph1}), runtimes (t_{Ph1}), and percent deviation from the best GRASP solutions, respectively. Column 7 gives the time to solve the allocation model with CPLEX. An optimality gap of 0% was used in all cases. Columns 8, 9 and 10 report the phase 2 results. In those computations, tabu search was allowed to run for a maximum of 50 iterations but was terminated earlier when no improving solution was found in 5 consecutive iterations. The 50-iteration limit was reached in only two instances. Column 9 gives the iteration on which the best phase 2 solution was found.

From Table 6.4, we see that the phase 1 solutions are 10.7% better on average than the best known solutions, and were obtained with significantly smaller runtimes. In fact, t_{Ph1} was 81% less than t_{grasp}, on average, whereas ϕ_{Ph1} was better than ϕ_{grasp} in all cases. The gap between ϕ_{Ph1} and ϕ_{grasp} ranged from 4 to 17%. As to be expected, ϕ_{Ph2} was smaller than ϕ_{Ph1} in all cases as well, providing an average improvement of 5.8%, as reported in the last column. For the GRASP, runtimes averaged 97.7 sec compared to 18.4 sec for phase 1 and 433.4 for phase 2. About 75% of t_{Ph2} resulted from calling CPLEX's LP solver to determine the optimal updated production quantities for each candidate move. If this option is omitted and only the *Production_Level_Adjustment_Algorithm* is used for this purpose, the GRASP is about 1% faster than our two-phase approach. The 5.8% improvement between phase 1 and phase 2, however, drops to approximately 4%.

Table 6.4. Comparison of solutions for problem instances with 50 customers and 20 periods

Prob. no.	(ϕ_{grasp}) Best GRASP solution	(t_{grasp}) Runtime (sec)	(ϕ_{Ph1}) Phase 1 solution	(t_{Ph1}) Runtime (sec)	[(ϕ_{Ph1} – ϕ_{grasp}) / ϕ_{grasp}]100%	(t_{ALLOC}) Alloc time (sec)	(ϕ_{Ph2}) Phase 2 solution	Iteration ϕ_{Ph2} found	(t_{Ph2}) Runtime (sec)	[(ϕ_{Ph2} – ϕ_{Ph1}) / ϕ_{Ph1}]100%
1	440,505	102.36	418,570	9.18	–5	5.6	399,125	25	226.45	–5
2	448,695	138.17	391,366	16.65	–13	13.0	373,581	29	385.55	–5
3	419,730	95.35	385,897	20.18	–8	16.25	353,058	22	234.16	–9
4	456,398	68.42	401,851	18.17	–12	17.0	361,309	19	216.41	–10
5	434,466	99.37	396,977	20.29	–8	13.89	365,035	16	725.57	–8
6	452,564	98.07	382,417	16.68	–15	14.6	368,082	50	467.14	–4
7	436,812	98.90	388,935	20.21	–11	14.97	369,963	30	404.73	–5
8	420,935	87.32	383,705	18.07	–9	15.91	370,822	30	329.34	–3
9	434,789	142.54	391,442	13.85	–10	15.25	379,401	26	392.90	–3
10	436,221	158.40	388,957	15.30	–11	15.37	370,805	25	408.05	–5
11	433,890	81.77	384,722	20.84	–11	17.83	357,107	30	316.56	–7
12	452,705	85.83	382,746	19.90	–15	16.78	355,199	45	585.90	–7
13	440,771	99.85	381,645	17.74	–13	14.58	366,547	20	340.97	–4
14	419,412	84.45	399,040	20.14	–5	17.23	364,115	35	335.42	–9
15	453,875	86.67	403,862	18.41	–11	14.43	367,659	30	555.09	–9
16	457,310	91.70	377,530	19.16	–17	15.68	360,534	25	493.80	–5
17	455,663	91.08	405,292	16.84	–11	13.33	398,442	20	153.88	–2
18	441,685	88.53	404,254	19.25	–8	16.0	368,600	39	574.6	–9
19	418,896	104.27	394,187	17.52	–6	14.07	377,073	32	488.67	–4
20	452,183	94.12	403,547	16.37	–11	13.43	372,141	39	635.94	–8
21	409,677	78.27	393,013	16.75	–4	13.61	374,743	16	160.18	–5
22	429,116	108.26	380,357	20.61	–11	17.61	347,449	50	1031.96	–9
23	443,184	106.99	387,351	16.52	–13	12.76	362,619	31	794.67	–6
24	426,113	101.40	388,221	20.34	–9	16.37	375,022	23	232.29	–3
25	462,245	86.99	386,524	19.45	–16	16.01	374,926	19	273.34	–3
26	442,029	82.15	397,620	20.29	–10	16.65	366,733	36	811.39	–8
27	444,695	85.69	385,085	20.30	–13	16.70	375,261	12	156.74	–3
28	449,894	187,46	388,354	22.84	–14	19.31	373,155	26	230.68	–4
29	461,555	93.93	400,043	19.09	–13	15.62	379,320	25	543.83	–5
30	434,006	93.73	397,217	20.00	–8	16.55	369,223	33	495.71	–7

The final steps of our methodology involved path relinking and solving the modified version of the allocation model (4-1) to obtain a lower bound ϕ_{LB}. The results for the 50-customer instances are reported in Table 6.5. Path relinking is a procedure used to explore the opportunity to improve the solution obtained from any methodology that provides multiple candidate solutions. It is based on the fact that paths between solutions give rise to neighborhoods that contain new solutions with attributes similar to those of the endpoints. Our algorithm explores paths between all pairs of *elite* solutions that were uncovered during tabu search. An elite solution is defined as the first improved solution following any unimproved solution.

The second column of Table 6.5 lists the number of elite solutions found, which averaged 4.4. Columns 3 – 5 give the path relinking solution, the corresponding runtime, and the gap with the phase 2 solution, respectively. In most cases, the latter is either negative or negligibly small calling into question the effort required to apply this procedure. The lower bound, ϕ_{LB}, obtained from the modified version of the allocation model is reported in column 6. The adjusted lower bound, ϕ_{ALB}, is presented in column 7 and was obtained by applying Eqs. (4), (6) – (8). On average, it showed an improvement over the lower bound, ϕ_{LB}, of 0.5% with a standard deviation of 1.23. The gap between the best solution and the adjusted lower bound averaged 12.8% and is shown in column 8. With respect to size, the lower bounding MIP contained 2,083 constraints and 3,183 variables of which 1,021 were binary. In all cases, a 500 sec limit was placed on CPLEX for these runs, which terminated with an optimality gap that averaged 2.2%.

The last three columns of Table 6.5 give the LP lower bound (ϕ_{LP}) for the full model (with the routing constraints), the corresponding solution times, and the gaps between the LP solution and the adjusted lower bound (ϕ_{ALB}) obtained from the modified allocation problem in 500 sec. The size of this gap, which was about 50%, implies that the solution to the LP relaxation by itself is not very useful.

Table 6.5. Path relinking and lower bound results for instances with 50 customers and 20 periods

Prob. no.	No. elites solns	(ϕ_{PR}) Solution from path relink	(t_{PR}) Time (sec)	[(ϕ_{PR} – ϕ_{Ph2}) / ϕ_{Ph2}]100%	(ϕ_{LB}) from LB model [a]	(ϕ_{ALB}) Adjusted LB	[(ϕ_{Best} – ϕ_{ALB}) / ϕ_{ALB}]100%	(ϕ_{LP}) LP soln	(t_{LP}) LP time (sec)	[(ϕ_{ALB} – ϕ_{LP}) / ϕ_{LP}]100%
1	8	398,795	180	–0.08	324,463	326,630	22	218,243	4.72	50
2	2	373,374	71	–0.06	325,469	326,324	14	219,206	4.69	49
3	6	353,058	460	0	326,365	329,607	7	217,697	5.56	51
4	6	361,176	300	–0.04	328,090	328,467	10	220,366	5.84	49
5	5	364,819	290	–0.06	324,451	325,044	12	220,178	3.8	48
6	3	368,082	222	0	329,875 [b]	332,449	11	221,852	5.3	50
7	9	369,963	720	0	326,712 [b]	328,176	13	218,911	4.08	50
8	1	370,822	*	0	324,958	325,893	14	216,969	6.92	50
9	2	379,379	36	–0.01	325,737	326,277	16	218,951	7.38	49
10	5	370,655	260	–0.04	325,846 [b]	326,483	14	217,392	4.92	50
11	7	354,025	540	–0.86	329,208	330,539	7	217,596	10.45	52
12	5	354,981	180	–0.06	320,151 [b]	326,316	9	215,297	6.33	52
13	3	365,432	208	–0.30	326,512	327,017	12	217,946	6.38	50
14	3	363,404	184	–0.20	321,141	323,498	12	216,514	4.74	49
15	4	367,659	56	0	325,857	326,339	13	219,531	5.69	49
16	1	360,534	*	0	326,324	328,384	10	216,828	5.33	51
17	5	398,442	120	0	328,513	329,996	21	221,842	5.94	49
18	5	368,533	189	–0.02	323,263	324,394	14	219,581	4.08	48
19	6	377,255	480	0.05	326,077	327,712	15	216,825	5.66	51
20	9	372,361	670	0.06	325,181	326,112	14	218,650	4.31	49
21	2	375,228	30	0.13	326,764	327,367	14	218,483	9.36	50
22	7	347,329	810	–0.03	322,818	324,015	7	214,260	6.61	51
23	4	362,619	220	0	326,345 [b]	327,824	11	219,942	5.44	49
24	2	375,609	100	0.16	320,055 [b]	321,520	17	214,406	4.98	50
25	3	374,682	163	–0.07	330,338 [b]	331,818	13	220,062	5.66	51
26	4	366,167	129	–0.15	331,722	334,062	10	220,034	5.16	52
27	1	375,261	*	0	321,615	322,816	16	216,087	5.14	49
28	3	373,464	120	0.08	323,233	324,047	15	215,604	4.12	50
29	5	379,320	255	0	334,784 [b]	336,159	13	223,175	5.11	51
30	6	370,012	420	0.21	331,837	335,212	10	218,849	6.33	53

* Only one elite solution

[a] Stop at 500 sec

[b] Lower bound model includes integer variables $y(t)$ (Lemma 1 not satisfied)

As discussed by Laporte (1992) and others, poor performance like this is due primarily to the weak relaxation associated with the subtour elimination constraints of the form $y_{jt} \le y_{it} - w_{it} + D_t^{\max}(1 - x_{ijt}),\ \forall\ i \in N, j \in N_0, t \in T$, where y_{it} is the load on the vehicle just after departing from customer i in period t and $D_t^{\max} = \min\left\{Q, \sum_{i \in N} \sum_{l=t}^{\tau} d_{il}\right\}$ is an upper bound on the load on any vehicle in period t. These constraints require the load on a vehicle to be monotonically decreasing during the delivery sequence and are rarely if ever binding in the LP solution. The corresponding MIP for the full model contained 54,063 constraints and 54,183 variables of which 51,021 were binary.

6.3 MEDIUM AND LARGE INSTANCES

The results for the 100-customer instances with 20 time periods each are presented in Tables 6.6 and 6.7. The column headings are identical to those of Tables 6.4 and 6.5, respectively. On average, the GRASP took 42.1 sec while tabu search took a total of 1,133.8 sec or 174% longer. The gap between the GRASP solution and the phase 1 solution obtained from the allocation model after running the VRP subroutine is identified in column 5 and represents roughly a 6.4% improvement. An additional 3.6% is realized in phase 2, giving a total improvement of 10%.

As seen in Table 6.7, path relinking takes about 465 sec and in only for problem no. 2 was an improvement realized. On average, the path relinking solutions were 1.2% worse than the phase 2 solutions. The modified allocation model had 4,083 constraints and 6,283 variables of which 2,021 were binary. A total of 850 sec was allotted for the computations. At termination, CPLEX exhibited an optimality gap of roughly 3.5%. The gap between the best feasible solution and the adjusted lower bound is shown in column 8 and averaged 22.9%. The data in the last three columns are as expected. The LP relaxation of the full model solves quickly with CPLEX but the gap between the solution, ϕ_{LP}, and the adjusted lower bound, ϕ_{ALB}, averaged 104%.

The results for the 200-customer instance with 20 time periods each exhibited the same patterns as the 100-customer instances.

Table 6.6. Comparison of solutions for problem instances with 100 customers and 20 periods

Prob. no.	(ϕ_{grasp}) Best GRASP solution	(t_{grasp}) Runtime (sec)	(ϕ_{Ph1}) Phase 1 solution	(t_{Ph1}) Runtime (sec)	[(ϕ_{Ph1} – ϕ_{grasp}) / ϕ_{grasp}]100%	(t_{ALLOC}) Alloc time (sec)	(ϕ_{Ph2}) Phase 2 solution	Iteration ϕ_{Ph2} found	(t_{Ph2}) Runtime (sec)	[(ϕ_{Ph2} – ϕ_{Ph1}) / ϕ_{Ph1}]100%
1	790,972	413.42	737,241	118.44	–7	96.50	711,671	21	1079	–3
2	782,906	506.19	734,043	143.27	–6	124.6	698,512	40	1035	–5
3	787,830	346.76	721,767	117.25	–8	95.43	683,270	40	1299	–5
4	779,847	486.51	741,904	119.51	–5	95.31	718,252	21	672	–3
5	796,176	439.06	748,370	118.33	–6	94.87	731,260	14	335	–2
6	793,216	349.02	758,969	141.54	–4	112.68	744,927	15	1143	–2
7	781,317	298.64	729,426	126.22	–7	105.39	695,728	36	1250	–5
8	780,884	499.19	729,542	149.80	–7	119.44	706,058	30	1008	–3
9	784,646	442.30	742,733	130.16	–5	108.22	705,035	39	1125	–5
10	790,156	378.54	727,431	143.40	–8	121.22	696,521	31	985	–4
11	787,596	393.90	734,881	126.43	–7	100.71	711,895	37	835	–3
12	785,170	369.25	730,530	134.65	–7	113.85	703,162	32	1129	–4
13	777,705	505.97	742,061	136.13	–5	111.88	721,066	22	599	–3
14	789,802	467.69	730,899	116.89	–7	95.30	698,548	35	1065	–4
15	790,132	527.02	732,911	157.43	–7	136.64	711,506	23	1139	–3
16	797,322	418.19	753,956	115.16	–5	91.20	714,873	37	1226	–5
17	799,843	520.11	729,914	125.23	–9	106.11	702,314	33	1218	–4
18	787,371	419.09	752,665	152.07	–4	126.81	720,238	32	720	–4
19	806,592	353.69	773,547	113.63	–4	85.87	748,734	36	1349	–3
20	809,340	403.09	748,000	123.80	–8	101.75	729,099	20	1131	–3
21	788,736	477.35	754,214	127.12	–4	102.28	738,746	14	544	–2
22	804,538	412.97	735,752	144.92	–9	123.60	702,849	36	998	–4
23	781,558	429.12	741,379	126.09	–5	101.82	712,717	23	1037	–4
24	798,428	416.88	758,063	142.42	–5	113.80	727,741	37	1380	–4
25	796,591	368.14	745,669	125.21	–6	102.85	725,869	30	1240	–3
26	791,514	403.69	724,380	124.91	–8	104.94	700,719	37	585	–3
27	773,662	495.51	709,640	118.37	–8	98.77	686,382	36	454	–3
28	780,492	416.89	724,556	143.13	–7	117.52	700,980	30	1190	–3
29	799,417	457.51	754,116	135.21	–6	110.31	725,030	32	993	–4
30	785,906	398.11	734,725	154.12	–7	134.15	698,942	32	1300	–5

Table 6.7. Path relinking and lower bound results for instances with 100 customers and 20 periods

Prob. no.	No. elites solns	(ϕ_{PR}) Solution from path relink	(t_{PR}) Time (sec)	[($\phi_{PR} - \phi_{Ph2}$) / ϕ_{Ph2}]100%	(ϕ_{LB}) from LB model [a]	(ϕ_{ALB}) Adjusted LB	[($\phi_{Best} - \phi_{ALB}$) / ϕ_{ALB}]100%	(ϕ_{LP}) LP soln	(t_{LP}) LP time (sec)	[($\phi_{ALB} - \phi_{LP}$) / ϕ_{LP}]100%
1	1	711,671	*	0.00	577,137[b]	582,090	22	281,684	24.18	107
2	4	694,694	240	–0.55	581,028[b]	584,847	19	284,882	22.02	105
3	2	693,600	80	1.51	578,440	581,317	18	283,960	36.05	105
4	1	718,252	*	0.00	571,459[b]	574,748	25	283,138	28.75	103
5	1	731,260	*	0.00	579,452	583,117	25	287,534	21.46	103
6	1	744,927	*	0.00	581,760	586,735	27	286,161	43.87	105
7	3	708,958	540	1.90	571,178	573,184	21	283,575	27.28	102
8	3	713,267	720	1.02	561,926[b]	564,981	25	278,847	34.79	103
9	2	713,779	360	1.24	580,835	583,295	21	286,051	24.54	104
10	2	710,427	270	2.00	573,489[b]	577,928	21	284,019	32.82	103
11	3	729,324	700	2.45	574,951[b]	580,831	23	284,048	20.95	104
12	6	712,832	550	1.38	577,810	581,643	21	285,148	20.62	104
13	5	728,004	650	0.96	568,636	572,655	26	281,696	23.24	103
14	2	705,304	500	0.97	578,958[b]	586,741	19	282,791	21.75	107
15	3	720,050	560	1.20	577,760	580,764	23	286,629	22.73	103
16	3	719,974	627	0.71	576,178	581,498	23	283,683	30.38	105
17	2	725,757	160	3.34	587,829[b]	591,956	19	289,069	11.40	105
18	2	731,189	140	1.52	577,616[b]	580,835	24	285,723	22.17	103
19	3	759,505	780	1.44	573,342	576,318	30	283,849	23.52	103
20	3	737,988	610	1.22	589,667[b]	593,649	23	290,279	33.05	105
21	1	738,746	*	0.00	584,901	590,280	25	289,479	94.78	104
22	2	719,634	400	2.39	574,965	577,327	22	285,958	25.14	102
23	1	712,717	*	0.00	567,679	569,334	25	281,909	23.82	102
24	3	738,178	720	1.43	574,446	577,506	26	285,820	20.57	102
25	6	739,029	590	1.81	573,169	576,299	26	284,631	21.70	102
26	3	707,568	520	0.98	567,682[b]	571,351	23	280,753	23.37	104
27	2	695,961	200	1.40	569,319	572,684	20	283,016	25.37	102
28	2	710,866	400	1.41	564,894	569,282	23	280,900	41.95	103
29	2	740,172	520	2.09	585,442	588,796	23	288,490	63.83	104
30	2	712,385	320	1.92	577,862[b]	582,331	20	286,316	47.36	103

* Only one elite solution

[a] Stop at 850 sec

[b] Lower bound model includes integer variables $y(t)$ (Lemma 1 not satisfied)

Table 6.8. Comparison of solutions for problem instances with 200 customers and 20 periods

Prob. no.	(ϕ_{grasp}) Best GRASP solution	(t_{grasp}) Runtime (sec)	(ϕ_{Ph1}) Phase 1 solution	(t_{Ph1}) Runtime (sec)	[(ϕ_{Ph1} − ϕ_{grasp}) / ϕ_{grasp}]100%	(t_{ALLOC}) Alloc time (sec)	(ϕ_{Ph2}) Phase 2 solution	Iteration ϕ_{Ph2} found	(t_{Ph2}) Runtime (sec)	[(ϕ_{Ph2} − ϕ_{Ph1}) / ϕ_{Ph1}]100%
1	1,075,528	2078.39	1,051,861	280	−2	179	1,030,684	20	2965	−2
2	1,070,340	1785.75	1,038,017	320	−3	233	1,024,558	11	1737	−1
3	1,070,505	1688.52	1,036,408	265	−3	131	1,036,282	4	1528	0
4	1,068,959	2194.75	1,067,202	289	0	174	1,057,654	14	2190	−1
5	1,060,220	1678.19	1,046,100	266	−1	189	1,031,422	18	2573	−1
6	1,065,700	1948.55	1,052,150	360	−1	204	1,033,233	19	1016	−2
7	1,091,538	1796.72	1,061,457	393	−3	206	1,043,536	35	1942	−2
8	1,060,164	2237.27	1,072,610	349	1	222	1,066,068	10	2978	−1
9	1,055,447	1554.42	1,047,509	321	−1	190	1,036,179	16	2454	−1
10	1,069,590	2090.38	1,057,926	377	−1	214	1,038,559	38	1855	−2
11	1,069,280	2464.75	1,054,845	388	−1	197	1,037,705	21	3428	−2
12	1,057,631	1909.58	1,052,501	397	−1	221	1,040,220	29	2288	−1
13	1,074,180	2087.02	1,075,537	483	0	233	1,063,024	13	2719	−1
14	1,076,460	2264.77	1,054,986	397	−2	216	1,041,786	21	3544	−1
15	1,065,340	1776.81	1,045,066	376	−2	155	1,029,908	25	2554	−2
16	1,067,550	2022.39	1,052,812	410	−1	231	1,033,656	27	3200	−2
17	1,067,007	1826.45	1,053,600	382	−1	187	1,027,433	31	2800	−3
18	1,095,350	1716.27	1,089,858	377	−1	172	1,063,306	29	4740	−2
19	1,063,445	1920.98	1,079,858	372	2	166	1,065,705	23	2797	−1
20	1,049,854	1910.66	1,057,882	360	1	148	1,034,195	37	2600	−2
21	1,055,436	2506.95	1,065,159	477	1	205	1,044,771	32	3900	−2
22	1,066,185	1781.39	1,059,796	369	−1	152	1,045,790	13	2822	−1
23	1,073,265	2201.73	1,033,344	459	−4	214	1,027,042	15	1797	−1
24	1,063,585	1999.14	1,078,993	381	1	151	1,051,610	37	3510	−3
25	1,054,230	1902.87	1,055,464	410	0	202	1,027,772	36	3000	−3
26	1,057,443	1685.31	1,059,502	335	0	129	1,044,315	26	2720	−1
27	1,076,798	1484	1,060,084	409	−2	213	1,047,267	12	2498	−1
28	1,054,225	1508.17	1,052,961	417	0	184	1,042,891	16	4200	−1
29	1,088,853	1463.53	1,040,105	409	−5	239	1,030,156	24	1947	−1
30	1,051,195	1613.92	1,051,980	325	0	155	1,035,703	21	2351	−2

Table 6.9. Path relinking and lower bound results for instances with 200 customers and 20 periods

Prob. no.	No. elites solns	(ϕ_{PR}) Solution from path relink	(t_{PR}) Time (sec)	[(ϕ_{PR} – ϕ_{Ph2}) / ϕ_{Ph2}]100%	(ϕ_{LB}) from LB model [a]	(ϕ_{ALB}) Adjusted LB	[(ϕ_{Best} – ϕ_{ALB}) / ϕ_{ALB}]100 %	(ϕ_{LP}) LP soln	(t_{LP}) LP time (sec)	[(ϕ_{ALB} – ϕ_{LP}) / ϕ_{LP}]100%
1	1	1,030,684	*	0	853,521	859,151	20	442,266	120.61	94
2	2	1,010,158	1250	–1.41	851,442[b]	856,489	18	440,625	121.81	94
3	2	1,016,681	1200	–1.89	843,605	847,677	20	438,910	123.88	93
4	4	1,042,854	2141	–1.40	852,993	858,620	21	440,440	132.46	95
5	3	1,023,680	2423	–0.75	853,415	858,986	19	440,789	78.34	95
6	2	1,025,262	2205	–0.77	859,458	863,514	19	443,041	75	95
7	3	1,038,746	2610	–0.46	855,652	859,755	21	442,400	111.27	94
8	1	1,066,068	*	0	856,121[b]	862,333	24	442,292	78.40	95
9	2	1,018,420	1250	–1.71	838,995	845,013	21	434,188	83.01	95
10	8	1,035,240	2625	–0.32	851,118	854,605	21	441,476	76.88	94
11	2	1,037,767	2730	0.01	861,136	866,478	20	444,524	104.01	95
12	5	1,035,350	2242	–0.47	856,766	861,683	20	443,012	105.69	95
13	1	1,063,024	*	0	851,622	855,760	24	442,760	78.43	93
14	2	1,024,491	1920	–1.66	856,951	860,535	19	443,699	103.96	94
15	6	1,026,787	2408	–0.30	848,998	855,899	20	440,104	75.98	94
16	2	1,043,917	1957	0.99	855,689	859,910	20	442,252	101.19	94
17	5	1,022,250	2018	–0.50	853,635	859,039	19	440,699	89.85	95
18	5	1,065,250	2242	0.18	854,938[b]	860,527	24	441,318	90.58	95
19	1	1,065,705	*	0	859,526	867,188	23	442,439	82.06	96
20	4	1,027,134	2425	–0.68	847,312	852,918	20	437,861	73.92	95
21	3	1,049,028	3003	0.41	849,209	853,126	22	440,479	80.13	94
22	1	1,045,790	*	0	848,907	854,260	22	439,190	67.17	95
23	4	1,034,198	2403	0.70	855,591	862,561	19	441,020	73.51	96
24	6	1,045,014	2908	–0.63	854,029	858,675	22	439,128	94.47	96
25	3	1,024,239	2601	–0.34	859,551	864,960	18	442,202	102.96	96
26	4	1,043,128	1673	–0.11	848,613[b]	855,543	22	438,277	93.42	95
27	2	1,030,753	2037	–1.58	863,295[b]	869,693	19	445,644	79.68	95
28	3	1,032,478	2909	–1.00	854,208	859,167	20	440,210	93.27	95
29	5	1,019,371	2179	–1.05	862,445	866,925	18	444,642	99.06	95
30	6	1,027,915	2400	–0.75	859,578	865,404	19	441,228	88.31	96

* Only one elite solution

[a] Stop at 1450 sec

[b] Lower bound model includes integer variables $y(t)$ (Lemma 1 not satisfied)

Table 6.8 presents the comparisons with GRASP and Table 6.9 presents the path relinking and allocation model results. The average improvement of tabu search over GRASP was 3% but average runtimes increased from 1,903 to 2,879 sec, or 51%. However, as the number of customers increases so do the objective function values, so small percentage reductions in cost often translate into be large reductions in absolute terms.

From Table 6.9, we see that path relinking solutions are on average 0.5% worse than the phase 2 solutions so it is doubtful whether the procedure is worthwhile, especially with runtimes averaging 2,230 sec. To compute the lower bound from the modified allocation model, CPLEX was allowed 1,450 sec. At the time, the average optimality gap was 3.6%, indicating the increased difficulty in solving the corresponding IP. The lower bound lemmas improved the results by 0.61% on average (not shown in table). The gap between the best solution found and the adjusted lower bound was approximately 20.5%, slightly better than for the 100-customer problems. For problem instances of this magnitude, these gaps are within an acceptable range.

Chapter 7

Solution Methodology – Branch and Price

An exact method based on branch and price (B&P) is discussed in this section. In simple terms, B&P combines (i) Dantzig-Wolfe (D-W) decomposition extended to accommodate integer variables, and (ii) standard branch and bound (B&B) (Vanderbeck 2000, Wolsey 1998). Below we outline how initial feasible solutions are obtained and then describe the principal components of our B&P algorithm.

7.1 INITIAL SOLUTIONS

There are two methods that we used to generate initial columns in the Master problem. In the first the columns were generated by using a simple algorithm where the deliveries to customers are made on the same periods where the demand occurs. In the second the initial columns were generated by using the solution from tabu search algorithm presented in Chapter 4.

7.2 COLUMN GENERATION

Like Lagrangian relaxation, D-W decomposition is most effective when the feasible region can be partitioned into a set of complicating or aggregate constraints and a set of easy or disaggregated constraints. For the current problem, it is natural to separate the production and inventory constraints (3-1b) – (3-1f), which cut across two time periods, from the routing and delivery constraints, which can be written separately for each time period. We use the former along with the objective function in (3-1a) to create the D-W master problem ($\mathcal{MP}$).

After removing the production and inventory constraints, we are left with (3-1g) – (3-1k) and the variable definitions, which conveniently decompose by time period giving τ subproblems. Points in the feasible region of subproblem t correspond to all feasible schedules on day t; where a schedule is a set of θ or fewer routes specifying the delivery

sequence and the amount delivered if a customer is serviced on that day. In the reformulated model, each column in $\mathcal{MP}$ corresponds to a feasible schedule for the n customers. In a solution, at most one column must be selected for each day.

Informally speaking, the idea behind B&P is to have the subproblems act as schedule generators guided by the values of the dual variables associated with the LP solution to a restricted $\mathcal{MP}$; i.e., one that contains only a subset of feasible schedules. At each major iteration, an optimality check is made by implicitly pricing out $\mathcal{MP}$. This is done by solving the subproblems to see whether one or more schedules can be identified that have a negative reduced cost. When added to the restricted $\mathcal{MP}$, such schedules in the form of columns will improve the current LP solution, which serves as a lower bound on the solution to the original IP (3-1). If all the reduced costs are nonnegative but fractional variables appear in the $\mathcal{MP}$ solution, B&B is performed. The approach is most effective when the LP solution to the restricted $\mathcal{MP}$ is found in the early stages of column generation.

7.2.1 Master problem

We first derive $\mathcal{MP}$ and then show how its solution is used to construct the objective function of the pricing subproblems. Let k be the column index and let the definitions of all previously introduced notation be the same. The following new symbols are used in the formulation of $\mathcal{MP}$.

Parameters and sets

c_t^k cost of the schedule for day t associated with column k

W_{it}^k amount delivered to customer i on day t in the schedule associated with column k

X_{ijt}^k mapping parameter; 1 if customer j is the immediate successor of customer i on day t in schedule associated with column k, 0 otherwise

$K(t)$ set of columns for day t

Decision variables

λ_t^k (binary) 1 if the schedule associated with column k on day t is selected, 0 otherwise

Master problem ($\mathcal{MP}$)

$$\phi_{\text{IP}} = \text{Minimize} \quad \sum_{t\in T}\sum_{k\in K(t)} c_t^k \lambda_t^k + \sum_{t\in T_0\setminus\{\tau\}} f_t z_t + \sum_{t\in T_0\setminus\{\tau\}} h^P I_t^P + \sum_{t\in T\setminus\{\tau\}}\sum_{i\in N} h_i^C I_{it}^C \tag{7-1a}$$

$$\text{subject to} \quad \sum_{k\in K(t)}\sum_{i\in N} W_{it}^k \lambda_t^k + I_t^p - I_{t-1}^P - p_t = 0, \ \forall\, t \in T \tag{7-1b}$$

$$\sum_{k\in K(t)} W_{it}^k \lambda_t^k - I_{it}^C + I_{i,t-1}^C = d_{it}, \ \forall\, i \in N, t \in T \tag{7-1c}$$

$$\sum_{k\in K(t)}\sum_{i\in N} W_{it}^k \lambda_t^k - I_{t-1}^P \le 0, \ \forall\, t \in T \tag{7-1d}$$

$$\sum_{k\in K(t)} \lambda_t^k \le 1, \ \forall\, t \in T \tag{7-1e}$$

$$p_t - C z_t \le 0, \ \forall\, t \in T_0 \setminus \{\tau\} \tag{7-1f}$$

$$p_0 \ge \sum_{i\in N}\left(q_{i1} - I_{i0}^C\right) \tag{7-1g}$$

$$\lambda_t^k \in \{0,1\}, \forall\, t \in T, k \in K(t); \ z_t \in \{0,1\}, p_t \ge 0, \forall\, t \in T_0 \setminus \{\tau\} \tag{7-1h}$$

$$0 \le I_t^P \le I_{\max}^P, \ 0 \le I_{it}^C \le I_{\max,i}^C, \forall\, i \in N, t \in T\setminus\{\tau\}; \ I_\tau^P = I_{i\tau}^C = 0,$$

$$\forall\, i \in N \tag{7-1i}$$

The objective function in (7-1a) minimizes the total production, inventory, and distribution cost over the planning horizon. The only change from (3-1a) is that the routing costs have been replaced with the cost of a schedule but these are really the same thing. Constraints (7-1b) – (7-1d), (7-1f) – (7-1g) are equivalent of (3-1b) – (3-1f), respectively, but written in terms of the schedule variables λ_t^k instead of the original variables (w_{it}, x_{ijt}). The parameters $\left(W_{it}^k, X_{ijt}^k\right)$ are derived from the solution of the pricing subproblems, which are presented next. The convexity constraint (7-1e) ensures that at most one delivery schedule is selected for each day.

When the integrality constraint on the λ_t^k variables in (7-1h) is relaxed, the solution to (7-1) provides a lower bound on the optimal solution to (3-1). With the D-W procedure, this bound is usually tighter than the bound obtained by solving the LP relaxation of (3-1) directly.

7.2.2 Pricing subproblems

The purpose of the subproblems is to identify "promising" columns for $\mathcal{MP}$. In the context of linear programming, a promising column is one with a negative reduced cost. When $\mathcal{MP}$ is re-solved with these columns included, the new solution should be an improvement over the current solution. If a promising column exists for day *t*, then it can be found by minimizing a generic representation of the reduced cost for that day. To formulate the objective function for subproblem *t*, let α_t^1 be the dual variable associated with the factory inventory constraint (7-1b), α_{it}^2 the dual variable associated with customer *i*'s inventory constraint (7-1c), α_t^3 the dual variable associated with the delivery constraint (7-1d), and α_t^4 the dual variable associated with the convexity constraint (7-1e). Only α_t^3 and α_t^4 are required to be nonnegative. For each day *t*, the reduced cost of column *k* in $\mathcal{MP}$ is

$$\bar{c}_t^k = c_t^k - \alpha_t^1 \sum_{i \in N} W_{it}^k - \sum_{i \in N} \alpha_{it}^2 W_{it}^k + \alpha_t^3 \sum_{i \in N} W_{it}^k + \alpha_t^4 \tag{7-2}$$

where $c_t^k = \sum_{i \in N_0} \sum_{j \in N_0} c_{ij} X_{ijt}^k$.

To put (7-2) into a form that is usable in the subproblems, the parameters $\left(W_{it}^k, X_{ijt}^k\right)$ must be expressed in terms of the original problem variables. Making the appropriate substitutions, collecting terms, and removing the column index *k*, gives

$$\bar{c}_t = \sum_{i \in N_0} \sum_{j \in N_0} c_{ij} x_{ijt} - \sum_{i \in N} \left(\alpha_t^1 + \alpha_{it}^2 - \alpha_t^3\right) w_{it} + \alpha_t^4 \tag{7-3}$$

The first term on the right-hand side of (7-3) represents the routing cost on day *t* while the remaining terms are the adjustments imposed by the $\mathcal{MP}$ dual values. In

formulating the subproblems, we retain all the routing and delivery constraints (3-1g) – (3-1k) of the original model.

Subproblem t ($\mathcal{SP}_t$)

$$\phi_t^{\mathrm{SP}} = \text{Minimize} \quad \sum_{i\in N_0}\sum_{j\in N_0} c_{ij}x_{ijt} - \sum_{i\in N}\left(\alpha_t^1 + \alpha_{it}^2 - \alpha_t^3\right)w_{it} + \alpha_t^4 \tag{7-4a}$$

$$\text{subject to} \quad \sum_{\substack{j\in N_0 \\ j\neq i}} x_{ijt} \le 1, \ \forall\, i \in N \tag{7-4b}$$

$$\sum_{\substack{i\in N_0 \\ i\neq j}} x_{ijt} = \sum_{\substack{i\in N_0 \\ i\neq j}} x_{jit}, \ \forall\, j \in N \tag{7-4c}$$

$$\sum_{j\in N} x_{0jt} \le \theta \tag{7-4d}$$

$$y_{jt} \le y_{it} - w_{it} + D_t^{\max}\left(1 - x_{ijt}\right), \ \forall\, i \in N, j \in N_0 \tag{7-4e}$$

$$w_{it} \le D_{it}^{\max} \sum_{j\in N_0} x_{ijt}, \ \forall\, i \in N \tag{7-4f}$$

$$x_{ijt} \in \{0, 1\},\ 0 \le y_{it} \le Q,\ w_{it} \ge 0 \text{ and integer}, \ \forall\, i \neq j \in N_0 \tag{7-4g}$$

In general, when the subproblems are all identical for a time decomposition, it is only necessary to solve one of them and then duplicate the results for all $t \in T$. If the parameter $D_{it}^{\max}$ were replaced by Q in (7-4f), then the constraints would be identical but the objective function (7-4a) would still differ from one day to the next because the dual variables $\left(\alpha_t^1, \alpha_{it}^2, \alpha_t^3, \alpha_t^4\right)$ are a function of t.

Although it is common to try to solve (7-4) to optimality, this may be too time consuming for large instances. All that is really needed, though, is a feasible solution whose objective function value in (7-4a) is negative. If one is found, then the corresponding column is added to the set $K(t)$ in $\mathcal{MP}$. Alternatively, if the computations are terminated before feasibility is achieved or if the smallest reduced cost of a feasible solution is nonnegative (this usually happens only when there are only a few columns in $\mathcal{MP}$), then no column is generated by the subproblem.

For the B&P approach to be valid, it is necessary to solve the LP relaxation of (7-1) to optimality. This means that all subproblems must eventually be solved to optimality to confirm that there are no remaining columns that price out negatively. The following proposition shows that only x_{ijt} are required to be integer variables in the subproblem.

Proposition 7.1 For any subproblem $\mathcal{SP}_t$, when the routing decision variables x_{ijt} are fixed to 0 or 1, there exists an optimal solution where the decision variables w_{it}, y_i are integral for all $i \in N$.

Proof. When x_{ijt} is fixed, let R_k, $k = 1,\ldots,\theta$ be the set of customers in route k, where $R_k = \{i_1^k, i_2^k, ..., i_{n_k}^k\}$ Figure 7.1 illustrates a case with $\theta = 2$. Here, the two routes are (0-1-3-5-7-0) and (0-2-4-6-0) with $x_{01t} = x_{13t} = x_{35t} = x_{57t} = x_{70t} = x_{02t} = x_{24t} = x_{46t} = x_{60t} = 1$ and all other $x_{ijt} = 0$. For the general case, we show that subproblem solution for the w_{it} and y_{it} variables can be obtained by solving a trivial knapsack problem for each route k.

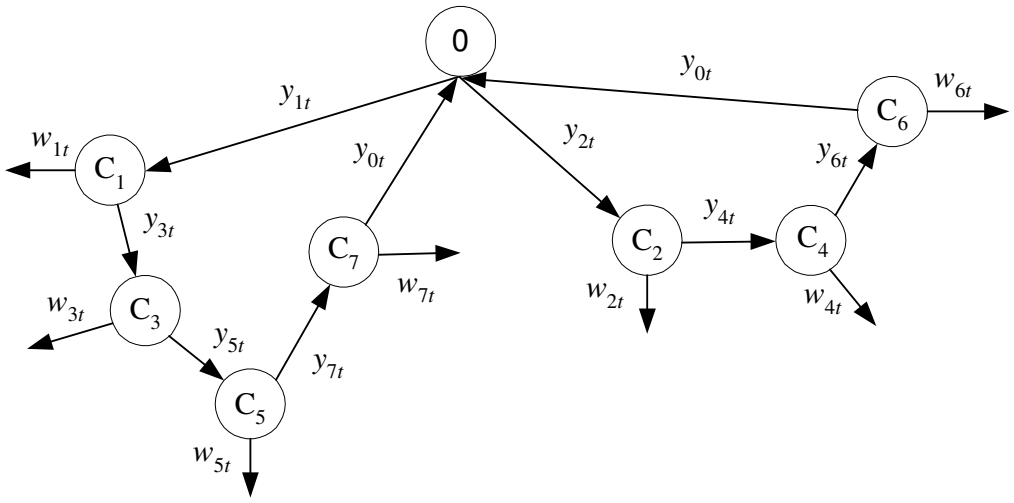

Figure 7.1. Example of subproblem when the x_{ijt} variables are fixed

To see this, let $\alpha_{it} \equiv \alpha_t^1 + \alpha_{it}^2 - \alpha_t^3$ be the coefficient of w_{it} in (7-4a) and for each R_k, order these coefficients from largest to smallest. Assume that the order is $i_1^k, i_2^k, ..., i_{n_k}^k$ for $k = 1,\ldots,\theta$. If $\alpha_{it} \le 0$ for any $i \in N$, then there is no advantage in making a delivery to customer i so $w_{it} = 0$. Now, for each R_k beginning with i_1^k, put $w_{it} =$ $\max\left(\min\left\{D_{it}^{\max}, Q_i\right\}, 0\right)$, where $Q_1 = Q$ and $Q_{i+1} = Q_i - w_{it}$ is the capacity available for customer i. Because $D_{it}^{\max}$ and Q are integral, w_{it} will be integral as well. Once the delivery quantities w_i^k are determined for $i \in R_k$, $k = 1,\ldots,\theta$, the corresponding values of y_i^k can be calculated recursively from $y_{i_{j+1}}^k = y_{i_j}^k - w_{i_j}^k$ for $j = 0,1,\ldots,n_k$, starting with $y_{i_0}^k =$ $\sum_{i \in R_k} w_{it}$ which is the load on the vehicle after it leaves the factory.

■

7.2.3 Branching strategies

A good branching strategy is the one that excludes the current solution, equally partitions the feasible region, and leads to pricing subproblems that are tractable. In general, branching strategies for 0-1 mixed integer formulations are based on fixing one or more variables at a time depending on whether special ordered sets (SOS) appear in the model.

With B&P, branching is initiated upon termination of D-W whenever integer variables in the $\mathcal{MP}$ have fractional values. In our case, there are two sets of binary variables in $\mathcal{MP}$, $\left\{\lambda_t^k : t = 1,..., \tau;\ k \in K(t)\right\}$ and $\left\{z_t : t = 1,..., \tau\right\}$, and even though each $\mathcal{SP}_t$ is solved as a MIP, only a few of the λ_t^k and z_t variables are likely to be integral. To move towards feasibility, we branch on both the master problem variables z_t and the subproblem variables x_{ijt} with priority given to the former. Note that branching on λ_t^k directly is not practical because it requires extensive modifications to $\mathcal{SP}_t$ at each iteration. When $\lambda_t^k = 1$, it is relatively easy to account for the solution associated with this variable in $\mathcal{SP}_t$ by fixing (w_{it}, x_{ijt}) at $\left(W_{it}^k, X_{ijt}^k\right)$ for all i and j. When $\lambda_t^k = 0$, however,

excluding this solution requires introducing an unmanageably large number of binary variables and logic constraints into the subproblem.

Two branching strategies are considered. In the first case, depth-first branching is performed on the z_t variables until no fractional z_t is found then we search for a fractional λ_t^k whose value is closest to 1, breaking ties arbitrarily, and then select a pair of customers, say i_1 and j_1, on the route associated with $\left(W_{i_1 t}^k, X_{i_1 j_1 t}^k\right)$, where the parameter $c_{i_1 j_1}$ is minimum, breaking ties arbitrarily. Standard depth-first search is used. On the left branch emanating from the current node, we set $x_{i_1 j_1 t} = 1$ and remove all columns from $\mathcal{MP}$ that do not include customers i_1 and j_1 in sequence on a route. In $\mathcal{SP}_t$, $x_{i_1 j_1 t}$ is also set to 1 and $w_{i_1 t}$ is determined by the solution from $\mathcal{SP}_t$. On the right branch, we set $x_{i_1 j_1 t} = 0$ and modify $\mathcal{MP}$ and $\mathcal{SP}_t$ accordingly, which includes setting $w_{i_1 t} = 0$. In general, a node is fathomed if the solution of $\mathcal{MP}$ is either no better than the incumbent, is integral, or is infeasible. Although this strategy is easy to implement, it is known to be inefficient because the feasible regions that result after branching are not balanced. The solution space of the problem after fixing $x_{i_1 j_1 t} = 0$ is almost the same as the solution space of the original problem. Consequently, we also implemented an SOS branching scheme.

In this approach, type 1 SOS branching is used to eliminate subsets of customers from being the immediate successor of some customer i in period t on a route. Implementation is based on the implied values of the subproblem variables (x_{ijt}) in $\mathcal{MP}$ along with Eq. (3-1g), which requires $\sum_{j \in N_0, j \neq i} x_{ijt} \leq 1$ for all $i \in N$, $t \in T$. The procedure is now described for an arbitrary node in the search tree.

For each period t in which fractional values of λ_t^k exist, we identify the customers who are scheduled to receive a delivery but are not on all routes $k \in K(t)$. This leads to the set

$$B = \left\{ (i,t) : \sum_{j \in N_0 \setminus \{i\}} \sum_{k \in K(t)} \lambda_t^k X_{ijt}^k < \sum_{k \in K(t)} \lambda_t^k,\ i = 1,...n;\ t = 1,...,\tau \right\}$$

If the choice for branching is not unique, then we select the candidate pair (i_1, t_1) as follows:

$$(i_1, t_1) \in \text{argmax} \left\{ \sum_{k \in K(i,t)} \lambda_t^k :\ (i,t) \in B \right\}$$

where $K(i,t) \subseteq K(t)$ is the subset of columns in period t for which customer i is on a route and $(i,t) \in B$. Remaining ties are broken arbitrarily. The rationale for the second criterion is to focus on the master problem variables, and indirectly, the subproblem variables that have the largest effect on the current solution.

Letting $N(i_1,t_1) \subset N_0$ be the set of customers who are the immediate successors of customer i_1 on routes in period t_1 where $\lambda_{t_1}^k$ is fractional, we partition on customer i_1 as follows. Let $\bar{N}$ be a subset of $N(i_1,t_1)$ in which $x_{i_1 j' t_1}$, $j' \in \bar{N}$, are free variables, and let $\bar{N}^1$ and $\bar{N}^2$ be a partition of the set $\bar{N}$ such that $\bar{N}^1 \cup \bar{N}^2 = \bar{N}$ and $\bar{N}^1 \cap \bar{N}^2 = \varnothing$. On one branch, we impose the constraint $\sum_{j \in \bar{N}^1, j \neq i_1} x_{i_1 j t_1} = 0$ which implies $\sum_{j \in \bar{N}^2, j \neq i_1} x_{i_1 j t_1} \leq 1$, and on the other we impose the constraint $\sum_{j \in \bar{N}^2, j \neq i_1} x_{i_1 j t_1} = 0$ which implies $\sum_{j \in \bar{N}^1, j \neq i_1} x_{i_1 j t_1} \leq 1$. To keep the two branches balanced, we try to maintain $\left|\bar{N}^1\right| \cong \left|\bar{N}^2\right|$ such that the number of free variables $x_{i_1 j' t_1}$ whose values are zero is approximately the same in both sets. Depending on the branch, $\mathcal{MP}$ is modified by removing all columns associated with the variables in either set $\bar{N}^1$ or $\bar{N}^2$, and $\mathcal{SP}_{t_1}$ is modified by setting the corresponding variables to zero.

When the summation $\sum_{j \neq i_1} x_{i_1 j t_1}$ has only one term, say, $x_{i_1 j_1 t_1}$, we revert to the standard strategy of creating the left branch with $x_{i_1 j_1 t_1} = 1$ and the right branch with $x_{i_1 j_1 t_1} = 0$. SOS branching for customer i in period t is exhausted when the set $\bar{N} = \varnothing$.

For the majority of routing and scheduling problems in which column generation is used, the original model has an assignment constraint in equality form. In our case, the equivalent constraint (3-1g) is written as an inequality because each customer need not be routed in each period. As a consequence, the convexity constraint (7-1e) in $\mathcal{MP}$ is also written as an inequality so fractional values of λ_t^k may exist in period t that do not sum to 1. In fact, there may be only one positive fractional value for λ_t^k in any period t in a relaxed solution so using the standard Ryan-Foster branching strategy is not recommended [see Ryan and Foster (1981) or Wolsey (1998) for a discussion of the procedure].

7.2.4 Information stored at each node

The efficient implementation of B&B requires critical information to be stored at each node of the search tree. Storing redundant information could result in excessive memory usage whereas storing insufficient information could result in an increase in runtime. To trade off between speed and memory usage we store the following information at each node.

a. Cost, route and delivery quantity associated with each column.
b. Parent of the current node.
c. Separate lists of subproblem variables x_{ijt} fixed to 1 and fixed to 0.
d. Separate lists of master problem variables z_t fixed to 1 and fixed to 0.
e. A parameter that identifies the first and the last column generated at the node.
f. List of columns where the associate master problem variable λ_t^k is set to 0.
g. Objective function value of the relaxed LP solution

The data structure used to store information for each node is a two-dimension array of what are termed "struct type variables" in Microsoft Visual Studio. The members of the struct type variable are those mentioned above. The indices of the array consist of the node id and period id. The nodes in B&B tree were created and managed using the MINTO 3.1 library.

Chapter 8

Algorithmic Issues

Two important issues arise when trying to solve the PIDRP with B&P that strongly affect algorithmic efficiency. The first concerns symmetry, where columns with the same delivery quantities but with reverse or permuted tours are generated at various nodes in the search tree. This is a common difficulty for routing and scheduling problems. The second involves the generation of null columns that specify a VRP solution without delivery quantities. In the following subsections, we describe how these issues are handled. Several important properties of the B&P algorithm are then summarized.

8.1 SYMMETRY

Recall that a column in $\mathcal{MP}$ represents a set of VRP tours. Because the order of the tours is not important and the reverse order of customers in a tour gives the same results, there are many symmetric solutions to the PIDRP. Difficulties occur when specific columns are excluded from model (7-1) during branching. New columns that have the same configuration but in reverse direction (symmetric columns) could still be generated and then included in $\mathcal{MP}$. As a result, the enumeration process can bog down when trying to eliminate symmetric columns.

The situation is illustrated in the Figure 8.1, which gives an example with two time periods and three customers. At the root node there are three columns generated for the first period and two for the second period. The numbers in each column represent the delivery quantity (w_{it}) to customer i in period t. For example, the column associated to λ_1^1 is $(2,0,8)^T$, where $w_{11} = 2$, $w_{21} = 0$ and $w_{31} = 8$. Based on our SOS branching rule, which selects the (i,t)-pair associated with the largest value of $\sum_{k \in K(i,t)} \lambda_t^k$ from the available elements in B, we get customer 3 in period 1. This leads to the following partition of the

feasible region: on the first branch we impose the constraint $x_{301} + x_{311} \leq 1$ while on the second node we impose $x_{321} \leq 1$. At node 1 then, we set $\lambda_1^2 = 0$ because we want to exclude the transition $3 \rightarrow 2$. At node 2 we set $\lambda_1^1 = 0$ to exclude transition $3 \rightarrow 1$.

When the modified subproblem is solved at node 1, the column that is fixed to zero $(0,2,5)^T$ can be regenerated with route 0-2-3-0. Similarly at node 2, the columns $(2,0,8)^T$ which is fixed at zero can be regenerated with route 0-1-3-0.

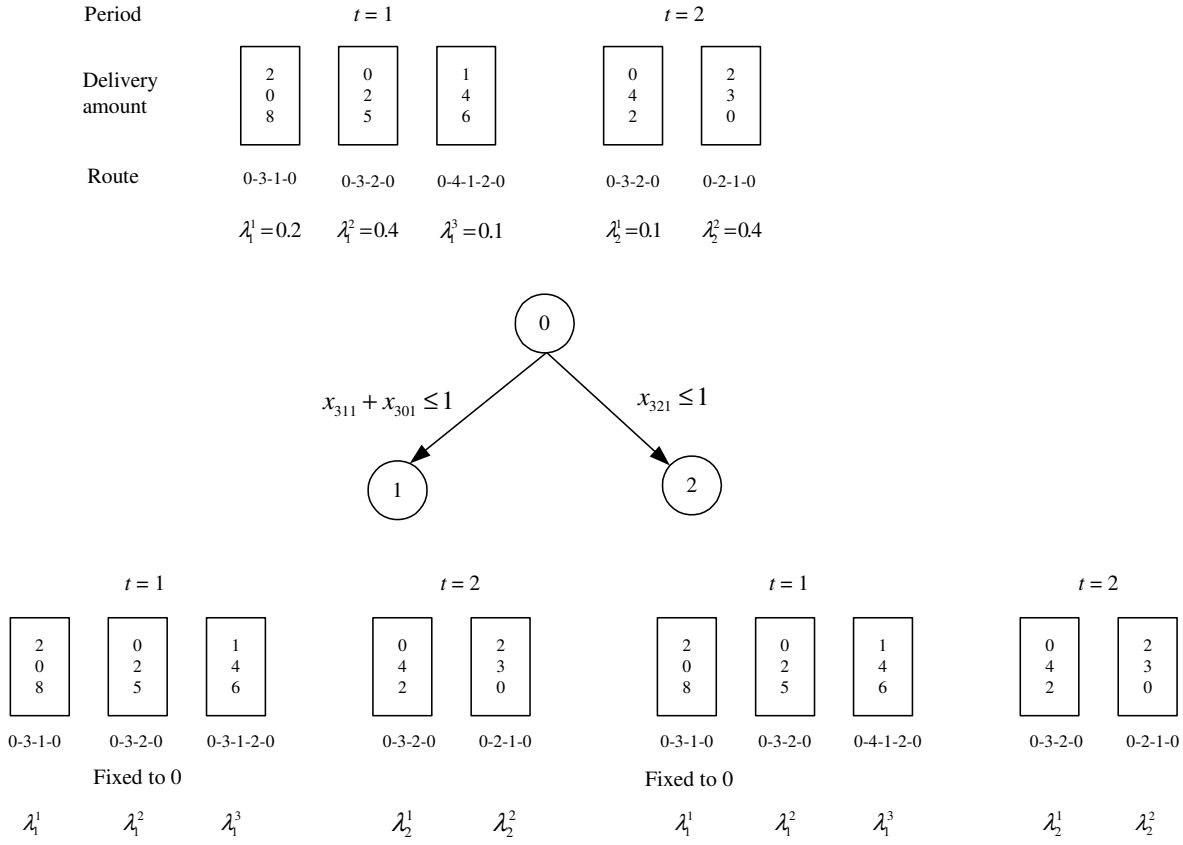

Figure 8.1. Example of symmetry problem

To prevent this situation from occurring, when columns are forced removed from $\mathcal{MP}$, we impose constraints that prevent their symmetric counterparts from being generated in the subproblem. For example, at node 1 when column $(0,2,5)^T$ is fixed to

zero, the constraint $x_{021} + x_{231} + x_{301} \leq 2$ is added to $\mathcal{SP}_1$ to prevent the symmetric route 0-2-3-0 from arising. Similar constraints are added to $\mathcal{SP}_1$ at node 2 to preclude route 0-1-3-0.

In general, it is inefficient to store both a column and its symmetric counterparts. Moreover, when $\mathcal{MP}$ contains more than one column that represents an equivalent solution, this could cause branching difficulties because when a column is forced to leave $\mathcal{MP}$ we need to make sure that its symmetric counterparts leave as well. To deal with this issue, we preprocess the generated routes before storing them. The IDs of the first and the last stops in a route are compared and if the ID of the first stop is less than the ID of the last stop, we store the route as it is; otherwise, we reverse the route and store it. For example, consider route 0-4-3-1-0 where the ID of the first stop is 4 and the ID of the last stop is 1. Because 4 > 1, the route is reversed and stored as 0-1-3-4-0.

8.2 BRANCHING PROCESS

There are several issues that need to be addressed to ensure the accuracy and efficiency of the branching process. The first involves the routing cost in each period. Because constraint (7-1e) in $\mathcal{MP}$, i.e., $\sum_{k \in K(t)} \lambda_t^k \leq 1, \forall\ t \in T$, is not written as an equality, when column generation terminates at a node, the routing cost in each period might not be correct. This will occur when $\sum_{k \in K(t)} \lambda_t^k < 1$ in some period t and the routes in $K(t)$ associated with positive λ_t^k are identical but have different delivery quantities. In this situation, the objection function value in $\mathcal{MP}$ for period t will be $\sum_{k \in K(t)} c_t^k \lambda_t^k$, which is less than the true value c_t^k.

To avoid under-representing the actual routing costs, new constraints of the form $\sum_{k \in K(t)} \lambda_t^k = \Lambda_t,\ 0 \leq \Lambda_t \leq 1$, were added to $\mathcal{MP}$ and used for branching when it was no longer possible to apply SOS branching. The choice of branching period, call it t^*, is determined as follows: $t^* = \arg\max \left\{ \sum_{k \in K(t)} \lambda_t^k, t \in T \right\}$. On one branch, Λ_{t^*} is forced to

1, on the other it is forced to 0. This guarantees that the final solution will have the correct routing cost in every period.

A second issue concerns the generation of null or zero-cost routes, which may occur when Λ_t is set to 1. By requiring $\sum_{k \in K(t)} \lambda_t^k = 1$ in period t, the corresponding subproblem is forced to generate a column. However, when $\alpha_t^1 + \alpha_{it}^2 - \alpha_t^3 \leq 0$ for all $i \in N$ in (7-4a), the subproblem solution will be $x_{ijt} = w_{it} = 0$ for all i and j, which translates into $c_t^k = 0$ in (7-1a). Putting the corresponding column into $\mathcal{MP}$ and solving the LP will yield a positive value for λ_t^k. Having zero-cost column in the solution, though, could lead to an incorrect objective value because zero-cost routes do not exist in reality. To avoid this situation, we add the constraint $\sum_{i \in N} w_{it} \geq 1$ to each $\mathcal{SP}_t$, thus ensuring that all routes generated will have non-zero cost.

Still, even when no further branching is possible, it may be that λ_t^k is not integral in some period t when $\sum_{k \in K(t)} \lambda_t^k = 1$ so it is not always straightforward to interpret a solution. In particular, there may be multiple columns that have the same delivery sequence but different delivery quantities. In this case, the following proposition shows that the optimal solution can be retrieved by taking a convex combination of columns to get $w_{it}^* = \sum_{k \in K(t)} \lambda_t^k W_{it}^k$, $i \in N$.

Proposition 8.1 When branching terminates, if there exists a period t in which λ_t^k, $k \in K(t)$, is not integral, then the optimal delivery quantity w_{it}^* in that period for each customer i is the convex combination of $\sum_{k \in K(t)} \lambda_t^k W_{it}^k$.

Proof. Be design, when branching terminates all columns in each period t have the same route and $\sum_{k \in K(t)} \lambda_t^k = 1$. When there is more than one value of λ_t^k, $k \in K(t)$, that is positive, the route associated with those columns can be replaced by a new column k^* such that $W_{it}^{k^*} = \sum_{k \in K(t)} \lambda_t^k W_{it}^k$, $i \in N$ and $c_t^{k^*} = \sum_{k \in K(t)} \lambda_t^k c_t^k$, where $c_t^{k^*} = c_t^k$ for all $k \in$

$K(t)$ because all route sequences are identical. Thus, we can replace the current set of columns in period t with $\lambda_t^{k^*} W_{it}^{k^*}$ without changing the objective function value. By following the same procedure for every period t where λ_t^k, $k \in K(t)$, is fractional we can get an equivalent solution for the delivery quantities w_{it}^* that is unique. ■

Figure 8.2 illustrates a nonintegral solution for an example with two periods and three customers. In period 1, $|K(1)| = 3$ with $\lambda_1^1 = 0.2$, $\lambda_1^2 = 0.4$, $\lambda_1^3 = 0.4$, and all three routes are 0-1-3-0; in period 2, $|K(2)| = 2$ with $\lambda_2^1 = 0.5$, $\lambda_2^2 = 0.5$ and both routes are 0-2-3-0.

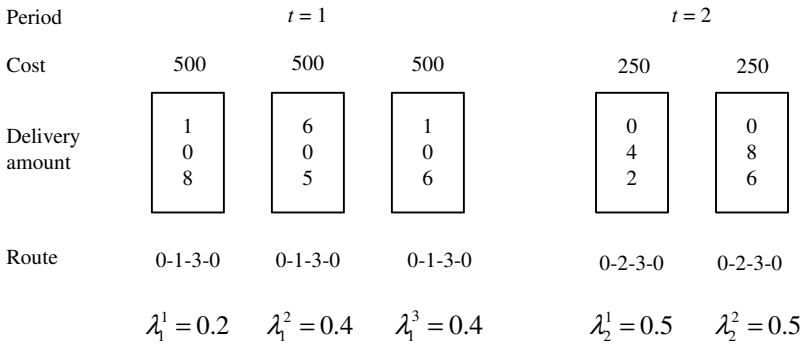

Figure 8.2. Example of how to interpret the solution when λ_t^k is not integral

Given that the current transportation cost is 500 for route 0-1-3-0, assume that $c_{01} = 100$, $c_{13} = 150$ and $c_{30} = 250$. Also, given a cost of 250 for route 0-2-3-0, let $c_{02} = 50$, $c_{23} = 150$ and $c_{30} = 50$. Based on Proposition 8.1, we can replace the three columns in period 1 with a new column k^* given by $(3,0,6)^{\mathrm{T}}$ and set $\lambda_1^{k^*} = 1$. The same applies to period 2 where the two columns can be replaced with a new column k^{**} $(0,6,4)^{\mathrm{T}}$ such that $\lambda_2^{k^{**}} = 1$. The solution in period 1 is to deliver three units to customer 1 and eight units to customer 3. The solution in period 2 is to deliver 6 units to customer 2 and four units to customer 4.

8.3 PROPERTIES OF B&P ALGORITHM

The following propositions show that the B&P algorithm is efficient with respect to column generation.

Proposition 8.2 Subproblem solutions never duplicate columns that have the same delivery amounts $\{w_{it} : i \in N\}$ but with reverse routes in any period *t*.

Proof. Using contradiction we show that by including the same column with reverse routing in $\mathcal{MP}$, the relaxed LP objective value cannot be improved. To begin, assume that column k^{**}, which has the same customers as an existing column k^{*} but with reverse routing, is generated from subproblem $\mathcal{SP}_t$ at the current iteration. Let $\lambda_t^{k^*,pre}$ be the multiplier value associated with column k^{*} at the previous iteration, and without loss of generality, assume that when the LP relaxation of $\mathcal{MP}$ is solved at the current iteration we get $\lambda_t^{k^*,cur}$ and $\lambda_t^{k^{**},cur}$ for columns k^{*} and k^{**}, respectively. Let K^{*} be the set of columns at the current iteration excluding columns k^{*} and k^{**}. Then the objective value of the LP relaxation at the previous iteration (z_{LP}^{pre}) is equal to $\sum_{k\in K^*} \lambda_t^{k,pre} c_t^k + \lambda_t^{k^*,pre} c_t^{k^*}$ while the LP relaxation at the current iteration (z_{LP}^{cur}) is equal to $\sum_{k\in K^*} \lambda_t^{k,cur} c_t^k + (\lambda_t^{k^*,cur} + \lambda_t^{k^{**},cur}) c_t^{k^*}$. Noting that $\sum_{k\in K^*} \lambda_t^{k,pre} + \lambda_t^{c^*,pre} \le 1$ and $\sum_{k\in K^*} \lambda_t^{k,cur} + (\lambda_t^{k^*,cur} + \lambda_t^{k^{**},cur}) \le 1$, we can set $\lambda_t^{k,pre} = \lambda_t^{k,cur}$, $\forall\ k \in K^{*}$, and $\lambda_t^{k^*,pre} = \lambda_t^{k^*,cur} + \lambda_t^{k^{**},cur}$ to get $z_{LP}^{cur} = z_{LP}^{pre}$. This contradicts the assumption that $z_{LP}^{cur} < z_{LP}^{pre}$, which is implied by the fact that the addition of column k^{**} to $\mathcal{MP}$ should lead to an improved solution.

■

Proposition 8.3 Only a finite number of columns with different delivery quantities $\{w_{it} : i \in N,\ t \in T\}$ and/or routes will be produced during column generation.

Proof. Each column in model (7-1) represents a point in the convex hull of the subproblem feasible region (7-4b) – (7-4g). From Proposition 7.1, we know that all the decision variables in $\mathcal{SP}_t$ will be integral in any feasible solution. Because x_{ijt} is binary, $0 \leq y_{it} \leq Q$ and $0 \leq w_{it} \leq D_{it}^{\max}$, the feasible region of $\mathcal{SP}_t$ is bounded for all $t \in T$. Thus, only a finite number of solutions exist.

■

Chapter 9

Enhanced Algorithmic Features

Based on preliminary test results given in Section 10.1, the size of PIDRP instances that can be solved with the B&P algorithm within 1 hour are limited to roughly 10 customers and 6 time periods. In this section, we describe several features for improving the efficiency of the algorithm. The first involves implementing a column generation heuristic to solve $\mathcal{SP}_t$. Next, we introduce a rounding heuristic that can be used to update the upper bound at selected nodes in the B&B tree. Finally, we present a model for determining periods in which at least one customer requires a delivery. This model is used in a preprocessing step to fix variables associated with the delivery decisions.

9.1 Heuristic Modifications

The amount of effort required to solve $\mathcal{SP}_t$ increases significantly as either the number of customers or the number of time periods increases. To ease this burden, we propose a two-step heuristic scheme for solving the subproblem (cf. Choi and Tcha 2007, Mourgaya and Vanderbeck 2007, Savelsbergh 1997). The first step involves developing a model for determining delivery quantities for each customer in each time period. Recall that $\mathcal{SP}_t$ is not a pure VRP. The second step involves finding actual routes in light of the current set of branching constraints. This is done with a VRP tabu search code that we adjust as necessary. Section 9.1.1 and 9.1.2 give the details of three models for determining delivery quantities. Section 9.2 describes how the branching constraints are handled in the VRP subroutine.

9.1.1 Heuristic model 1 for determining delivery quantities

VRP algorithms require that the delivery quantities to be known, even for cases in which visiting a customer is optional as in the prize collecting version of the problem. One way to estimate these quantities is to solve a linear program whose objective function

coefficients are based, in part, on the current values of the master problem dual variables. Once again let $\alpha_{it} = \alpha_t^1 + \alpha_{it}^2 - \alpha_t^3$ be the coefficient associated with w_{it} in the subproblem objective (7-4a) and let c_i be the cost of visiting customer i. In reality, c_i can only be determined after the routing decisions are made and is equal to $\Sigma_j c_{ij}x_{itj}$ in period t. Alternatively, we can estimate its value by, say, $f_{it}^C = 2c_{i0}$ for all i and t, which is the cost of a round trip between the factory and customer i.

For $\alpha_{it} > 0$, there is some value in making a delivery to i in period t. The ratio $\alpha_{it} / \left(c_i + \alpha_t^4\right)$ gives the relative value for each customer. Ideally, we would like this ratio to be greater than or equal to 1 but as noted, it is impossible to know the value of c_i a priori. For our purposes, then, it is sufficient to consider all i for which $\alpha_{it} > 0$ and to remove the constant term α_t^4 from the ratio. The first heuristic model (H_1) is as follows.

$$\text{Maximize} \quad \sum_{i=1}^{n}\sum_{t=1}^{\tau} \max\{0, \alpha_{it} / c_i\} w_{it} \tag{9-1a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} w_{it} \le \theta Q, \quad t = 1,\ldots,\tau \tag{9-1b}$$

$$\sum_{l=1}^{t} w_{il} \ge \sum_{l=1}^{t} d_{il}, \quad i = 1,\ldots,n;\ t = 1,\ldots,\tau \tag{9-1c}$$

$$0 \le w_{it} \le D_{it}^{\max}, \ i = 1,\ldots,n;\ t = 1,\ldots,\tau \tag{9-1d}$$

The objective function (9-1a) is designed to maximize the value of the items delivered to all customers over the planning horizon. The coefficients in (9-1a) will change at each $\mathcal{MP}$ iteration. Constraints (9-1b) limit the delivery quantities in each period to the fleet capacity. Constraints (9-1c) ensure that a sufficient amount is delivered to each customer over the planning horizon to meet their demand in each period. Bounds are placed on the delivery variables in (9-1d). The upper bound limits the amount delivered to any customer i in period t to the minimum of the capacity of a vehicle or the remaining demand for that customer from t through τ. It does not restrict the total amount delivered over the planning horizon to be equal to the total demand.

Model (9-1) ignores the routing aspect of the PIDRP so there may be some periods in which it is not possible to service all those customers whose w_{it} values are greater than 0. This issue can be addressed in two ways. First, we can reduce the right-hand side of (9-1b) by some fraction to compensate for the routing constraints. Alternatively, the VRP code discussed presently could be modified to accommodate infeasible solutions by heavily penalizing violations of (9-1b) rather than prohibiting them. If the capacity of a vehicle was exceeded in a solution, then the individual delivery quantities w_{it} could be reduced incrementally until the violation was removed, say, starting with the customer whose coefficient $\alpha_{it} > 0$ was the smallest. During the reduction process, it would be necessary to ensure that (9-1c) remained satisfied and that the reduced cost of the solution remained negative.

Proposition 9.1 The feasible region given by constraints (9-1b) – (9-1d) is a relaxation of the feasible region of the original problem.

Proof. The load variable in the original problem, y_{it}, is bounded by Q in each period t. Letting $\ddot{y}_t = \max\{y_{it} : i = 1,...,n\}$ and forming the product $\theta\ddot{y}_t$ gives an upper bound on the delivery quantity in period t which is equivalent to (9-1b). Constraints (9-1c) can be derived by substituting out the inventory variables, I_{it}^C, in Eq. (3-1c) of the original model and noting that $I_{it}^C \geq 0$ for all i and t. Finally, (9-1d) is a relaxation of (3-1k) in the original model. ■

Proposition 9.2 An optimal solution exists to model (9-1) in which w_{it} is integral for all $i \in N$ and $t \in T$ when all the problem data are integral.

Proof. We show that if any of the w_{it} are fractional, the solution cannot be optimal. In the simplest case in which there is at most one fractional value of w_{it} in period t, it is always possible to increase its value to $\lceil w_{it} \rceil$ without violating any of the constraints because all the data are assumed to be integral. The objective function would then increase by

$\left(\lceil w_{it} \rceil - w_{it}\right) \times \left(\max\{0, \alpha_{it} / c_i\}\right)$ in period *t*. For the case in which, say, $w_{i_1 t}$ and $w_{i_2 t}$ are fractional in period *t*, let us assume without loss of generality that $\max\{0, \alpha_{i_1 t} / c_{i_1}\} \geq \max\{0, \alpha_{i_2 t} / c_{i_2}\}$. Now, increasing and decreasing the delivery quantities to customers i_1 and i_2 by some arbitrarily small amount ε, respectively, produces a marginal improvement in the objective function value without violating either constraints (9-1b) or (9-1d). The increase in $w_{i_1 t}$ cannot lead to a violation of (9-1c) nor can the reduction of $w_{i_2 t}$. The latter claim follows from the fact that (9-1c) must be nonbinding for those constraints in which $w_{i_2 t}$ appears due to the fact that the right-hand sides are integral. All other cases are generalizations of these two and so the same arguments apply to show that either the solutions cannot be optimal or that an equivalent integral solution can be constructed.

■

9.1.2 Heuristic model 2and 3 for determining delivery quantities

Our inability to specify exactly the cost of visiting customer *i* in period *t* may distort the selection of delivery quantities w_{it} in the solution to (9-1a) – (9-1d). A more accurate model can be formulated by reintroducing the routing variable x_{ijt}. In doing so, we limit the augmented feasible region to include only the assignment constraints to ensure that optimal solutions can be obtained quickly.

The augmented model, call it H_2, is

$$\text{Maximize} \quad -\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=1}^{\tau} c_{ij} x_{ijt} + \sum_{i=1}^{n}\sum_{t=1}^{\tau} \alpha_{it} w_{it} \tag{9-2a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} w_{it} \leq \theta Q, \quad t = 1,\ldots,\tau \tag{9-2b}$$

$$\sum_{l=1}^{t} w_{il} \geq \sum_{l=1}^{t} d_{il}, \quad i = 1,\ldots,n;\ t = 1,\ldots,\tau \tag{9-2c}$$

$$0 \le w_{it} \le D_{it}^{\max} \sum_{\substack{j=0 \\ j \ne i}}^{n} x_{ijt}, \quad i = 1,\ldots,n;\ t = 1,\ldots,\tau \tag{9-2d}$$

$$\sum_{\substack{j=0 \\ j \ne i}}^{n} x_{ijt} \le 1, \quad i = 1,\ldots,n;\ t = 1,\ldots,\tau \tag{9-2e}$$

$$\sum_{\substack{i=0 \\ i \ne j}}^{n} x_{ijt} = \sum_{\substack{i=0 \\ i \ne j}}^{n} x_{jit}, \quad j = 1,\ldots,n;\ t = 1,\ldots,\tau \tag{9-2f}$$

$$\sum_{j=1}^{n} x_{0jt} \le \theta, \quad t = 1,\ldots,\tau \tag{9-2g}$$

$$x_{ijt} \in \{0,1\},\ \forall\ i, j, t \tag{9-2h}$$

The objective function (9-2a) now weighs the cost of making a delivery to customer i in period t from the corresponding benefit. Constraints (9-2b) – (9-2d) are the same as their counterparts in (9-1) except that the upper bound in (9-2d) has been modified to prevent delivery to customer i if no routing assignment is made. Constraints (9-2e) ensure that at each customer has at most one successor, while constraints (9-2f) guarantee continuity of flow. The final inequalities (9-2g) ensure that at most θ vehicles leave the depot in period t. In (9-2h), binary restrictions are placed on the flow variables.

A solution to (9-2) is a weak approximation to the collective solutions of the τ subproblems. The primary differences are that the individual vehicle capacity restrictions are not maintained in (9-2) and that a solution may have subtours in each time period that do not include the factory. Nevertheless, the advantage of model (9-2) is that it should be much easier to solve than the subproblems, and like model (9-1), it should provide reasonable values for the delivery quantities w_{it}, for all $i \in N$ and $t \in T$, that can serve as input to the VRP subroutine.

The third model proposed for determining delivery quantities, call it H_3, is similar to H_2 (9-2) except constraint (9-2c) is removed. The advantage of H_3 is that the runtimes are likely to be much less than for H_2, since (9-2a) – (9-2b), (9-2d) – (9-2h) decomposes into τ separate problems. The computational results for the different heuristics can be found in Section 10.3

9.2 HANDLING BRANCHING CONSTRAINTS IN VRP SUBROUTINE

With SOS branching, restrictions are placed on the x_{ijt} variables in $\mathcal{SP}_t$ such that one or more of them is set to either 0 or 1 at each node in the search tree. We use the tabu search code developed by Carlton and Barnes (1996) for solving the corresponding VRPs, modifying the distance matrix to account for these restrictions. To handle the case where x_{ijt} is fixed to 0, the distance matrix coefficient c_{ij} is set to a large positive number M. For the case where x_{ijt} is fixed to 1, the values of c_{ik}, $k \in N_0$ and $k \neq i$, are likewise set to M. The following section describes how the value of M can be selected without affecting the VRP solution.

9.2.1 Big M cost in VRP subroutine

Although it was possible to modify the VRP code to exclude or include certain arcs in a solution, doing so is not a straightforward matter. Instead, we chose to modify the arc costs to obtain the same results; however, care is needed in selecting the value of M. If M is too small, the algorithm might converge to an infeasible solution that has a lower cost than any feasible solution. As a consequence, an appropriate value for M needs to be determined. We begin by deriving a lower bound on infeasible solutions and an upper bound on feasible solutions.

Proposition 9.3 Let I be a set of customers that must be visited in some period ($w_i > 0 : i \in I$), let $c_i^{\min} = \min\{c_{ij} : j \in N_0\}$ for all $i \in N_0$, and let $f^{\max} = \max\{c_i^{\min} : i \in I \cup \{0\}\}$. Then the lowest bound on the objective function value for any infeasible solution (a solution with at least one arc cost set to M) is

$$\sum_{i \in I} c_i^{\min} + \left\lceil \sum_{i \in I} w_i / Q \right\rceil c_0^{\min} - f^{\max} + M .$$

Proof. Visiting all the customers in I incurs a cost of at least $\sum_{i \in I} c_i^{\min}$. The minimal number of vehicles that must be used for this purpose is $\left\lceil \sum_{i \in I} w_i / Q \right\rceil$, which translates into a cost of at least $\left\lceil \sum_{i \in I} w_i / Q \right\rceil c_0^{\min}$. A solution is considered infeasible if it contains an arc that should be excluded; such and arc will have a cost of M. In order to get the

infeasible route with the lowest cost, we replace the cost on the arc that gives $f^{\max}$ with M. As a result, a lower bound on the cost for any infeasible solution is $\sum_{i \in I} c_i^{\min} + \left\lceil \sum_{i \in I} w_i / Q \right\rceil c_0^{\min} - f^{\max} + M$. Note that setting the cost of more than one arc to M will result in a route with a higher cost because we want $M > f^{\max}$. When $M < f^{\max}$ it is possible to have an infeasible solution with a lower cost than a feasible solution – an unwanted situation.

■

Proposition 9.4 Let I be a set of customers that must be visited ($w_i > 0 : i \in I$) and let $c_i^{\max} = \max\{c_{ij} : j \in N_0\}$. Then an upper bound on the cost of any feasible solution is $\sum_{i \in I} c_i^{\max} + |I| c_0^{\max} + \sum_{i \in N \setminus I} c_i^{\max}$.

Proof. The maximum cost of visiting customers in I is $\sum_{i \in I} c_i^{\max}$. In the worst case, one vehicle will be needed for each such costumer giving a cost of $|I| c_0^{\max}$. The last term $\sum_{i \in N \setminus I} c_i^{\max}$ represents the cost of having to visit other customers that are not in I in order to avoid using infeasible arcs (those with costs set to M). In the worst case, all customers in $N \setminus I$ must be visited giving the stated bound. ■

Proposition 9.5 The value of M must be greater than M^* given below to guarantee that there is no feasible solution that has a higher cost than any infeasible solution.

$$M^* = \sum_{i \in N} c_i^{\max} + |I| c_0^{\max} - \left\{ \sum_{i \in I} c_i^{\min} + \left\lceil \sum_{i \in I} w_i / Q \right\rceil c_0^{\min} - f^{\max} \right\}$$

Proof. To validate the bound $M > M^*$, two cases will be considered. The first involves constructing a lower bound for infeasible solutions (routes that have at least one arc with cost set to M), and the second involves constructing an upper bound for feasible solutions. From Proposition 9.3, let

$$Cost_1 = \sum_{i \in I} c_i^{\min} + \left\lceil \sum_{i \in I} w_i / Q \right\rceil c_0^{\min} - f^{\max} + M$$

be a lower bound on the cost of an infeasible solution, and from Proposition 9.4 let

$$Cost_2 = \sum_{i \in N} c_i^{\max} + |I| c_0^{\max}$$

be an upper bound on the cost of a feasible solution. Then M^* is the minimal value of M such that $Cost_1 > Cost_2$; that is,

$$\sum_{i \in I} c_i^{\min} + \left\lceil \sum_{i \in I} w_i / Q \right\rceil c_0^{\min} - f^{\max} + M > \sum_{i \in N} c_i^{\max} + |I| c_0^{\max}.$$

Solving for M gives the above result. ■

9.2.2 Dealing with symmetry in subproblem heuristic

When $\mathcal{SP}_t$ is not solved exactly, we are not able to eliminate symmetric columns by using the approach discussed in Section 8.1 because it is not practical to modify the VRP code to accommodate the necessary constraints. Alternatively, we propose to adjust the SOS branching scheme to reduce the effects of symmetry. The idea is that when selecting the (i,t) for SOS branching, to give preference to customers that already have some restrictions imposed on their routes in period t. Although this does not eliminate the difficulties associated with symmetry completely, it mitigates its effect.

9.3 ROUNDING HEURISTIC ALGORITHM

When solving large-scale MIPs obtaining feasible solutions is critical because they provide upper bounds that can be used to speed convergence regardless of the algorithm, as well as offering a course of action should it be impractical to reduce the optimality gap to zero. In our B&P implementation, we periodically try to convert fractional solutions at a particular node in the search tree to a feasible solution with a rounding heuristic. The methodology uses the tabu search algorithm presented in Chapter 4.

Step 1. From a fractional $\mathcal{MP}$ solution, calculate the total delivery quantity ψ_{it}^* for customer i in period t as follows:

$$\psi_{it}^* = \sum_{k \in K(i,t)} W_{it}^k \lambda_t^k$$

where $K(i,t) \subseteq K(t)$ is the subset of columns in period t for which a delivery to customer i is indicated.

Step 2. For w_{it} fixed at ψ_{it}^* for all i and t, determine whether the remaining inventory and production constraints are feasible by solving a modified version of model (3-1). If feasible, then, starting with $\left(\psi_{it}^* : \forall\ i,t\right)$ and the computed values $\left(\left(I_{it}^C, I_t^P\right) : \forall\ i,t\right)$, run the tabu search algorithm to obtain an improved solution at the current node and update the incumbent. If the input data are not feasible, then the upper bound remains the same.

9.4 MODEL FOR DETERMINING PERIODS WITH DELIVERY REQUIREMENT

The convexity constraints (7-1e) in $\mathcal{MP}$ are written as inequalities rather than as equalities because there is no stated requirement to make a delivery in any particular time period except perhaps the first. As a consequence, a solution to the restricted $\mathcal{MP}$ may have fractional values of λ_t^k in period t whose sum is less than 1. This implies that only a fraction of a route is selected. During branch and bound, this issue is ultimately resolved with our SOS branching rule and does not cause any difficulties.

Nevertheless, when $\sum_{k \in K(t)} \lambda_t^k$ is fractional, $\mathcal{MP}$ produce infeasible solutions, thus introducing a degree of inefficiency in the computations. One way to strengthen the convexity constraint is to identify periods in which at least one customer must be visited. The following sequence of $\tau - 1$ optimization problems patterned after model (9-1) can be used for this purpose.

Problem P_t ; $t = 2,\ldots,\tau$

$$\zeta_t = \text{Minimize} \quad \sum_{i=1}^{n} w_{it} \tag{9-3a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} w_{i\gamma} \le \theta Q\ ,\ \gamma = 2,\ldots,\tau \tag{9-3b}$$

$$\sum_{l=1}^{\gamma} w_{il} \ge \sum_{l=1}^{\gamma} d_{il}\ ,\ \ i = 1,\ldots,n;\ \gamma = 2,\ldots,\tau \tag{9-3c}$$

$$0 \le w_{i\gamma} \le D_{i\gamma}^{\max}, \ i = 1,\ldots,n; \ \gamma = 2,\ldots,\tau \tag{9-3d}$$

$$\sum_{i=1}^{n} w_{i,\gamma-1} \ge \zeta_{\gamma-1}, \ \gamma = 2,\ldots, t \tag{9-3e}$$

where $\zeta_1 = \sum_{i=1}^{n} d_{i1}$

The objective function (9-3a) determines the minimum quantity that must be delivered in period t subject to the conditions that no shortages occur in any period over the planning horizon and that the load on the θ vehicles doesn't violate the capacity of the fleet. These conditions are ensured by (9-3b) – (9-3d) which duplicate (9-1b) – (9-1d), except that now the index t has been replaced with γ to avoid confusion with sequence index t. The new constraints (9-3e) require that the aggregate minimum delivery quantities computed for periods prior to t be maintained in all subsequent periods; however, the individual quantities w_{it} may change.

A solution to P_t with $\zeta_t > 0$ indicates that a delivery must take place in period t so the corresponding convexity constraint in $\mathcal{MP}$ can be written as an equality. Nevertheless, the fact that the feasible region (9-3b) – (9-3e) is a relaxation of the routing constraints implies that there may exist periods in which $\zeta_t = 0$ but a delivery is required.

Proposition 9.6 There are no feasible solutions to the original problem (3-1) in which a delivery must take place in period t when the solution of model (9-3) returns $\zeta_t = 0$.

Proof. From Proposition 9.1, we know that constraints (9-3b) – (9-3d) are a relaxation of the original feasible region. Constraint (9-3e) states that in period t all previously found minimum delivery quantities computed in periods $1,\ldots,t-1$ cannot be reduced, which is a feasibility condition for the PIDRP. If a feasible solution to the PIDRP existed in which $\Sigma_i w_{it} > 0$ or some period t where $\zeta_t = 0$, that solution would also be feasible to (9-3) and hence contradicts the optimality of ζ_t.

■

Model (9-3) is designed to schedule deliveries as early as possible in periods preceding the current period t to achieve a ζ_t as small as possible without regard to production, inventory or routing considerations. When $\zeta_t = 0$, it is often possible to avoid deliveries in subsequent periods by increasing deliveries in periods 1,…,t which constraint (9-3e) permits. This observation has implications for branching. If we set $\sum_{k \in K(t_1)} \lambda_{t_1}^k = 0$ in period t_1, this condition might require that deliveries be scheduled in some previous or subsequent period t_2 so we could correspondingly set $\sum_{k \in K(t_2)} \lambda_{t_2}^k = 1$ at the current node in the B&B tree.

Chapter 10

Computational Result – Branch and Price

In this section we report several computational results. Section 10.1 gives the results for the basic B&P algorithm whereas Sections 10.2 and 10.3 discuss the performance of the B&P heuristic for a range of configurations and data sets.

For the initial experiments summarized in Section 10.1, we randomly generated 7 instances on a 100×100 grid. The first step was to fix the number of customers $n \in \{5, 10\}$ and the number of time periods $\tau \in \{2, 4, 6, 8\}$ by selecting combinations from these sets. The remaining parameter values were generated using the same procedure provided in Section 6.1. The data sets used in the experiments described in Sections 10.2 is the same as those used in Section 10.1. In Section 10.3, we report the performance of the B&P heuristic when a series of benchmark data sets were used for testing.

All computations were performed on a 2.53 GHz processor with 512 MB of RAM. The optimization models were implemented in Java Netbean 4.1 and linked to the CPLEX 8.1 libraries. CPU times were obtained through both CPLEX and the time function in Java. The B&P algorithm and its components were coded using Microsoft Visual Studio .NET 7.0. The MINTO 3.1 library was used to facilitate the implementation and management of the search tree and the various components of branch and price.

10.1 Results from basic B&P algorithm

Table 10.1 provides a summary of the results obtained by running the basic B&P algorithm on seven randomly generated test problems for a maximum of 30 minutes each. The algorithmic components are as described in Section 7, except that the initial columns were obtained by using a simple procedure where the deliveries to customers are made in

the same periods in which the demand occurs. Results for the case where the algorithm was initiated with the tabu search solution are presented in Section 10.3.

In Table 10.1, columns 2 and 3 give the number of time periods τ and the number of customers n, respectively, for each instance. Columns 4 and 5 report the solutions obtained by solving model (3-1) with CPLEX (ϕ_{cplex}) and the corresponding runtimes (t_{cplex}). Columns 6 and 7 summarize the solutions from B&P ($\phi_{\text{B\&P}}$) and their runtimes ($t_{\text{B\&P}}$). The last column reports the percentage gap between these two solutions, which is calculated as: $((\phi_{\text{B\&P}} - \phi_{\text{cplex}}) / \phi_{\text{cplex}})100\%$. The computations were halted if the optimal solution was not found within 1800 seconds or the computer's memory limit was exceeded. Because it is relatively expensive to solve $\mathcal{SP}_t$ exactly, or even until a feasible solution is found with negative objective function value, only one column was added to $\mathcal{MP}$ at a time.

Table 10.1. Solutions from B&P algorithm

Problem no.	τ	n	(ϕ_{cplex}) CPLEX solution	(t_{cplex}) CPLEX runtime (sec)	($\phi_{\text{B\&P}}$) B&P solution	($t_{\text{B\&P}}$) B&P time (sec)	$[(\phi_{\text{B\&P}} - \phi_{\text{cplex}}) / \phi_{\text{cplex}}]$ 100%
1	2	5	21,961	8	21,961	34.3	0
2	2	10	38,776	62	38,776	1075	0
3	4	5	24,416	>1800	24,416	>1800	0
4	4	10	48,287	>1800	52,321	>1800	8.4
5	6	5	44,377	>1800	44,831	>1800	1.02
6	6	10	71,546	>1800	70,804	>1800	–0.29
7	8	5	54,558	>1800	56,073	>1800	3.77

From Table 10.1 we can see that when $\tau \geq 4$ and $n\tau \geq 20$ the runtimes of both CPLEX and the B&P algorithm increase dramatically. This was to be expected because the number of binary variables and constraints in models (3-1) and (7-1) grow quadratically with the number of customers. However, when comparing the results, the

B&P solution is approximately 1.84% higher on average than the solution provided by CPLEX. This indicates that the B&P algorithm is somewhat inefficient for relatively small instances, a general characteristic that can be understood by looking at the detailed statistics in Table 10.2.

In the table column 4 gives the LP solution (ϕ_{LP}) for the PIDRP, while columns 5 and 6 report the root node solution ϕ_{Root} and its gap with respect to ϕ_{LP}. Column 7 reports the total number of nodes processed, column 8 lists the number of columns generated at the root node, and column 9 gives the total number of columns generated. The last column indicates the total amount of time in seconds spent to solve the subproblems.

Table 10.2. Detail of results from B&P algorithm

Problem no.	τ	n	(ϕ_{LP}) LP solution	(ϕ_{Root}) Root node solution	[($\phi_{Root} - \phi_{LP}$) / ϕ_{LP}] 100%	($\eta_{B\&P}$) Number of nodes processed	(κ_{Root}) Number of columns at root node	($\kappa_{B\&P}$) Total number of columns	(t_{Sub}) Total subproblem time (sec)
1	2	5	21,181	21,218	0.17	61	18	133	21.6
2	2	10	38,275	38,303	0.07	773	75	1967	949
3	4	5	22,140	22,369	1.03	2245	45	2712	1431
4	4	10	45,723	46,012	0.63	695	183	1290	1510
5	6	5	41,024	41,116	0.2	903	91	1675	1620
6	6	10	67,136	67,393	0.38	693	281	1236	1670
7	8	5	49,013	49,248	0.48	1975	114	1975	1723

On average, the root node solution ϕ_{Root} is approximately 0.43% higher than the LP solution ϕ_{LP}, which shows that the lower bounds provided by column generation at the root node are quite weak. Also, the total amount of time devoted to generating the columns is 1275 seconds, on average, which is approximately 88% of the total runtime. This is not surprising because the complexity of the subproblem increases significantly when the number of customers (n) increases.

The primary conclusion that can be drawn from these results is that CPLEX is likely to outperform B&P on relatively small instances. The advantage of B&P comes when solving larger instances with the enhanced features discussed in Section 9. Before presenting those results, we examine the performance of the three heuristic models proposed for determining delivery quantities during column generation.

10.2 COMPUTATIONAL RESULTS FOR HEURISTIC MODEL FOR DETERMINING DELIVERY QUANTITIES

In Sections 9.1.1 and 9.1.2, we presented three models (H_1, H_2, H_3) that can be used in conjunction with a VRP code to heuristically solve the pricing subproblems (7-4a) – (7-4g). To determine the most suitable model, we performed several tests comparing results for each of the three heuristics at the root node with those obtained from the standard column generation approach. Table 10.3 highlights the output statistics for the standard approach. Columns 4 and 5 give the solution at the root node (ϕ_{Root}) and the deviation of the root node solution from the LP solution, respectively. Columns 6 and 7 report the number of columns generated and the total runtime.

Table 10.3. Results using column generation to solve the PIDRP at the root node

Problem no.	τ	n	(ϕ_{Root}) Root node solution	[($\phi_{Root} - \phi_{LP}$) / ϕ_{LP}] 100%	(κ_{Root}) Number of columns	(t_{Root}) Total runtime (sec)
1	2	5	21,218	0.17	18	3.27
2	2	10	38,303	0.07	75	25.95
3	4	5	22,369	1.03	45	3.97
4	4	10	46,012	0.63	183	238
5	6	5	41,116	0.2	91	12.64
6	6	10	67,393	0.38	281	688.05
7	8	5	49,248	0.48	114	22.23

In all cases, the root node solutions are slightly higher than the LP solutions, with an average deviation of 0.43%. For those instances with the same value of τ, the

deviations increase moderately with n. For example, problem nos. 3 and 4 both have 4 time periods with 5 and 10 customers, respectively. While n increases by a factor of 2, the deviation goes up by 36%. As we have seen, the number of columns and the total runtime increase with τ and n. For the test problems with the same number of time periods, the total runtimes change significantly when the number of customers increases. On average, when n increases by a factor of 2 the runtimes increase by factor of 40, which is directly attributable to the additional time required to solve each $\mathcal{SP}_t$.

Tables 10.4, 10.5 and 10.6 report the computational results when H_1, H_2 and H_3 are used, respectively, to solve the PIDRP at the root node. Columns 4 and 5 give the root node solutions (ϕ_{H_Root}) and their deviation from the solutions obtained with standard column generation (ϕ_{Root}). The number of columns generated at the root node (κ_{H_Root}) is reported in column 6. The runtime (t_{H_Root}) and the deviation from the runtimes of standard column generation (t_{Root}) are reported in columns 7 and 8, respectively.

From the tables we can see that H_1 gives the smallest solutions times, t_{H1_Root}, on average, which are 76.67% less than the average values of t_{Root}; however, the deviation of ϕ_{H1_Root} from ϕ_{Root} is approximately 71.31% higher. This calls into question the effectiveness of H_1 because good heuristics should give solutions relatively close to the optimum, ϕ_{Root}. The deviation of ϕ_{H3_Root} and ϕ_{H2_Root} from ϕ_{Root} are 0.03% and 0.09% on average, implying that H_3 gives the best solutions at the root node. Also, since t_{H3_Root} is 48.23% lower than t_{H2_Root} on average, and the number of columns generated (κ_{H3_Root}) is roughly the same as κ_{H2_Root}, H_3 is arguably the best choice of the three. The solutions provided by H_3 have the smallest deviations from ϕ_{Root} with 71.17% shorter runtimes when t_{H3_Root} is compared to t_{Root}, and 23.67% fewer columns when κ_{H3_Root} is compared to κ_{Root}.

When implementing column generation, it is common to follow one of two strategies: (i) at each iteration begin solving the subproblems and insert into $\mathcal{MP}$ only the first column that prices out negatively, or (ii) find as many columns as possible and insert all of them into $\mathcal{MP}$. The previous results are for the first strategy. To determine which

approach gives better performance with respect to the heuristics, we reran the test problems using the second strategy.

Table 10.4. Results using column generation heuristic H_1 to solve the PIDRP at the root node

Problem no.	τ	n	(ϕ_{H1_Root}) Root node solution	[($\phi_{H1\_Root} - \phi_{Root}$) / ϕ_{Root}] 100%	(κ_{H1_Root}) Number of columns	(t_{H1_Root}) Total runtime (sec)	[($t_{H1\_Root} - t_{Root}$) / t_{Root}] 100%)
1	2	5	21,223	0.02	20	2.98	–8.87
2	2	10	38,322	0.05	31	1.95	–92.49
3	4	5	31,099	39.03	7	1.16	–70.78
4	4	10	62,896	36.69	9	4.14	–98.26
5	6	5	70,976	72.62	11	3.19	–74.76
6	6	10	126,423	87.59	19	3	–99.56
7	8	5	178,832	263.13	8	1.78	–91.99

Table 10.5. Results using column generation heuristic H_2 to solve the PIDRP at the root node

Problem no.	τ	n	(ϕ_{H2_Root}) Root node solution	[($\phi_{H2\_Root} - \phi_{Root}$) / ϕ_{Root}] 100%	(κ_{H2_Root}) Number of columns	(t_{H2_Root}) Total runtime (sec)	[($t_{H2\_Root} - t_{Root}$) / t_{Root}] 100%)
1	2	5	21,223	0.02	16	1.22	–62.7
2	2	10	38,322	0.05	39	2.84	–89.06
3	4	5	22,394	0.11	43	3.83	–3.53
4	4	10	46,151	0.3	172	16.14	–93.22
5	6	5	41,121	0.01	118	7.56	–40.19
6	6	10	67,457	0.09	168	48.06	–93.02
7	8	5	49,259	0.02	98	19.34	–13

Table 10.6. Results using column generation heuristic H_3 to solve the PIDRP at the root node

Problem no.	τ	n	(ϕ_{H3_Root}) Root node solution	[($\phi_{H3\_Root} - \phi_{Root}$) / ϕ_{Root}] 100%	(κ_{H3_Root}) Number of columns	(t_{H3_Root}) Total runtime (sec)	[($t_{H3\_Root} - t_{Root}$) / t_{Root}] 100%)
1	2	5	21,218	0.00	10	1.11	–66.06
2	2	10	38,305	0.01	22	3.38	–86.97
3	4	5	22,369	0.00	34	2.48	–37.53
4	4	10	46,094	0.18	119	9.55	–95.99
5	6	5	41,122	0.01	56	4.67	–63.05
6	6	10	67,419	0.04	273	19.21	–97.21
7	8	5	49,250	0.00	102	10.81	–51.37

Tables 10.7, 10.8 and 10.9 report the results when H_1, H_2 and H_3 are respectively used to generate multiple columns at the root node. From the tables we can see that, on average, $t^*_{H1\_Root}$ (the total runtime), is slightly less than t_{H1_Root}, whereas $t^*_{H2\_Root}$ is 54.67% less than t_{H2_Root}, and $t^*_{H3\_Root}$ is 56.56% less than t_{H3_Root}. For H_2 and H_3, the explanation for these results is that $\kappa^*_{H2\_Root}$ and $\kappa^*_{H3\_Root}$ decrease with respect to κ_{H2_Root} and κ_{H3_Root} so the master problem LPs are smaller. For H_1, the number of generated columns $\kappa^*_{H1\_Root}$ is greater, on average, than κ_{H1_Root}, but the number of LPs solved is less. In addition, we see that the deviation from ϕ_{Root} is reduced from 71.31% for ϕ_{H1_Root} to about 0.33% for $\phi^*_{H1\_Root}$, which shows that generating multiple columns provides a large improvement in the performance of H_1. However, the deviation of $\phi^*_{H1\_Root}$ from ϕ_{Root} is still relatively high compared to the deviations of $\phi^*_{H3\_Root}$ and $\phi^*_{H2\_Root}$ from ϕ_{Root}, which are only 0.04% and 0.08%, respectively.

Table 10.7. Results using H_1 with multi-column generation to solve the PIDRP at the root node

Problem no.	τ	n	$(\phi^*_{H1\_Root})$ Root node solution	$[(\phi^*_{H1\_Root} - \phi_{Root}) / \phi_{Root}]$ 100%	$(\kappa^*_{H1\_Root})$ Number of columns	$(t^*_{H1\_Root})$ Total runtime (sec)	$[(t^*_{H1\_Root} - t_{Root}) / t_{Root}]$ 100%)
1	2	5	21,223	0.02	16	2.86	–12.54
2	2	10	38,322	0.05	31	2.03	–92.18
3	4	5	22,369	0.00	35	1.8	–54.66
4	4	10	46,771	1.65	30	1.97	–99.17
5	6	5	41,145	0.07	40	1.95	–84.57
6	6	10	67,719	0.48	86	3.84	–99.44
7	8	5	49,267	0.04	81	3.61	–83.76

Table 10.8. Results using H_2 with multi-column generation to solve the PIDRP at the root node

Problem no.	τ	n	$(\phi^*_{H2\_Root})$ Root node solution	$[(\phi^*_{H2\_Root} - \phi_{Root}) / \phi_{Root}]$ 100%	$(\kappa^*_{H2\_Root})$ Number of columns	$(t^*_{H2\_Root})$ Total runtime (sec)	$[(t^*_{H2\_Root} - t_{Root}) / t_{Root}]$ 100%)
1	2	5	21,223	0.02	13	3.02	–7.65
2	2	10	38,322	0.05	32	3.86	–85.13
3	4	5	22,369	0.00	36	2.38	–40.05
4	4	10	46,147	0.29	86	11.38	–95.22
5	6	5	41,124	0.02	51	3.48	–72.47
6	6	10	67,482	0.13	123	15.83	–97.70
7	8	5	49,268	0.04	76	4.89	–78.00

Table 10.9. Results using H_3 with multi-column generation to solve the PIDRP at the root node

Problem no.	τ	n	$(\phi^*_{H3\_Root})$ Root node solution	$[(\phi^*_{H3\_Root} - \phi_{Root}) / \phi_{Root}]$ 100%	$(\kappa^*_{H3\_Root})$ Number of columns	$(t^*_{H3\_Root})$ Total runtime (sec)	$[(t^*_{H3\_Root} - t_{Root}) / t_{Root}]$ 100%)
1	2	5	21,218	0.00	16	2.39	–26.91
2	2	10	38,312	0.02	52	4.76	–81.66
3	4	5	22,369	0.00	32	1.5	–62.22
4	4	10	46,087	0.16	85	3.75	–98.42
5	6	5	41,122	0.01	53	2.22	–82.44
6	6	10	67,441	0.07	118	5.25	–99.24
7	8	5	49,250	0.00	68	2.36	–89.38

This analysis indicates that H_2 and H_3 are superior to H_1 with respect to solution quality, and that the deviation of $\phi^*_{H3\_Root}$ and $\phi^*_{H2\_Root}$ from ϕ_{Root} is about the same as the deviation of ϕ_{H3_Root} and ϕ_{H2_Root} from ϕ_{Root}. Looking at the number of columns generated, we see that $\kappa^*_{H1\_Root}$ is 203% larger than κ_{H1_Root}; however, $\kappa^*_{H2\_Root}$ and $\kappa^*_{H3\_Root}$ are 36.24% and 31.17% less than κ_{H2_Root} and κ_{H3_Root}, respectively. When we consider only H_1 and H_2 the results show that generating multiple columns at each iteration is beneficial in term of shortening runtimes, reducing the overall size of $\mathcal{MP}$, and providing good solution quality.

The analysis also shows that H_2 and H_3 with multi-column generation outperform H_1 and exact column generation with respect to runtime. In addition, their average statistics for solution quality and the number of columns generated are about the same. A significant difference arises, though, when we consider runtime: $t^*_{H3\_Root}$ is 50.39% less than $t^*_{H2\_Root}$, which suggests that overall, H_3 with multi-column generation is the best choice. On average, H_3 gives good quality solutions, leads to relatively small master problems, and runs quickly.

Nevertheless, when H_2 and H_3 with multi-column generation were incorporated in the B&P algorithm we found that H_2 was more efficient as measured by the size of the search tree, runtime, and solution quality. B&P could only solve small instances when H_3 was used. Table 10.10 reports test results comparing H_2 with H_3 on the first two problem instances.

Table 10.10. Performance comparison of H_2 and H_3 with multi-column generation when incorporated in the B&P heuristic

Problem no.	τ	n	$\phi_{H2\_B\&P}$	$\phi_{H3\_B\&P}$	($o_{H2\_B\&P}$) Number of nodes	($o_{H3\_B\&P}$) Number of nodes	($\kappa_{H2\_B\&P}$) Number of columns	($\kappa_{H3\_B\&P}$) Number of columns	($t_{H2\_B\&P}$) Total runtime (sec)	($t_{H3\_B\&P}$) Total runtime (sec)
1	2	5	21,961	24,760	7	486	18	420	14	53
2	2	10	38,776	48,489	5	502	38	445	23	77

In Table 10.10 columns 4 and 5 give the respective B&P solutions when H_2 and H_3 were used. Columns 6 and 7 give the corresponding number of nodes in the search tree at termination, and columns 8 and 9 give the number of columns generated. The last two columns report the total runtime. From these statistics, we see that on average the number of nodes ($o_{H3\_B\&P}$) and the number of columns generated ($\kappa_{H3\_B\&P}$) are more than 1000% larger than $o_{H2\_B\&P}$ and $\kappa_{H2\_B\&P}$. Also, the total runtime $t_{H3\_B\&P}$ is 251% larger than $t_{H2\_B\&P}$, while solution $\phi_{H3\_B\&P}$ is 19% higher than $\phi_{H2\_B\&P}$.

The relative inefficiency of H_3 when compared to H_2 is due to the fact that the columns generated by H_3 are not necessarily feasible with respect to customer demand over the planning horizon τ. Although this is not integral to $\mathcal{SP}_t$, it proved to be beneficial. The advantage of H_2 is that it takes this consolidation requirement into account with constraints (9-2c) so the columns provided by H_2 at each iteration include all customers who must be serviced to ensure feasibility. Without this constraint the size of the B&B tree increases by an order of magnitude. General speaking, it takes a significant amount of branching effort to consolidate deliveries into the same column.

In summary, H_2 demonstrated the best performance when implemented with the B&P algorithm and will be used in the experiments in the following section.

10.3 RESULTS FROM ENHANCED B&P ALGORITHM

In this section, we investigate the various options that we proposed for the B&P algorithm, and provide extensive test results for the more promising combinations. Section 10.3.1 details the results associated with the following four enhanced features: (i) initializing $\mathcal{MP}$ with columns obtained from the tabu search solution, (ii) implementing a modified partitioning scheme which gives preference to branching on $\sum_{k \in K(t)} \lambda_t^k = \Lambda_t$ over the SOS variables, (iii) using a preprocessor to determine those periods that have delivery requirements, and (iv) periodically calling the rounding heuristic to obtain feasible solutions during B&B. In all runs, H_2 was used to determine delivery quantities. In Section 10.3.2 we report computational results for the B&P heuristic for 100 benchmark instances when the best settings found from the first set of experiments are used.

10.3.1 Results with different configurations

To gain a better understanding of the effectiveness of the four options, we conducted four experiments. A maximum of 1800 seconds was allowed for the computations in all runs. Table 10.11 presents the results obtained by solving the PIDRP with the B&P heuristic using the best settings found so far (H_2 for the subproblem; branching order $z_t \rightarrow \Lambda_t \rightarrow$ SOS(x_{ijt})). The subscript "E" is used instead of "H" to distinguish the output statistics from those presented in the previous section. The values in Table 10.11 provide the baseline for judging the configurations tested.

Tables 10.12 – 10.15 report similar statistics for the different experiments. Columns 4, 5 and 6 are the solutions from CPLEX (ϕ_{cplex}), tabu search (ϕ_{TS}) and B&P heuristic ($\phi_{\text{B\&P\_E1}}$, $\phi_{\text{B\&P\_E2}}$, $\phi_{\text{B\&P\_E3}}$, or $\phi_{\text{B\&P\_E4}}$), respectively. Column 7 lists the gap between these solutions and ϕ_{cplex}. Columns 8 and 9 report the total number of nodes in the search tree and the total number of columns generated. The last two columns give the total runtimes for the B&P heuristic and the corresponding values for the subproblems.

Table 10.11. Results using the B&P heuristic to solve the PIDRP

Problem no.	τ	n	(ϕ_{cplex}) CPLEX solution	($\phi_{B\&P\_E}$) B&P solution	[($\phi_{B\&P\_E} - \phi_{cplex}$) / ϕ_{cplex}] 100%	($\eta_{B\&P\_E}$) Number of nodes processed	($\kappa_{B\&P\_E}$) Number of columns	($t_{B\&P\_E}$) Total runtime (sec)	(t_{Sub_E}) Total subproblem time (sec)	(n_{Master_E}) Total $\mathcal{MP}$s solved
1	2	5	21,961	21,961	0	7	18	13.69	14.23	19
2	2	10	38,776	38,776	0	5	38	23.39	17.82	30
3	4	5	24,416	25,770	5.55	1981	465	1315	1132.6	2327
4	4	10	48,287	56,258	16.51	853	1238	>1800	1631.22	2035
5	6	5	44,377	57,509	29.59	1239	443	>1800	1519	1621
6	6	10	71,546	92,622	29.46	769	738	>1800	1721.34	1391
7	8	5	54,558	75,316	38.05	1928	440	1534	1289.12	2294

The first enhancement tested involved initializing $\mathcal{MP}$ with columns obtained from solving the PIDRP with our tabu search algorithm. The results are presented in Table 10.12. Compared to the basic results in Tables 10.11, $\phi_{B\&P\_E1}$ is 13.62% less than $\phi_{B\&P\_E}$ on average. In addition, the number of nodes processed ($\eta_{B\&P\_E1}$) decreased by 21.62%, the number of columns generated ($\kappa_{B\&P\_E1}$) increases by 110.68%, and the total runtime ($t_{B\&P\_E1}$) decreased by 22.04%. These statistics indicate that algorithmic efficiency can be improved considerably by initiating column generation with good solutions.

Next, we experimented with reversing the branching order by giving priority to Λ_t rather than the SOS variables. The results are reported in Table 10.13, and when compared to the statistics on solution quality in Table 10.12 indicate that, on average, $\phi_{B\&P\_E2}$ is only 0.06% less than $\phi_{B\&P\_E1}$. However, the number of nodes processed ($\eta_{B\&P\_E2}$) decreased by 48.67%, the number of columns generated ($\kappa_{B\&P\_E2,}$) decreased by 10.13% and the total runtime ($t_{B\&P\_E2}$) decreased by 35.47%. By implication then, adopting this branching scheme reduces the computational burden but has little effect on solution quality.

Table 10.12. Results using the B&P heuristic to solve the PIDRP with initial columns from tabu search

Problem no.	τ	n	(ϕ_{cplex}) CPLEX solution	(ϕ_{TS}) Tabu solution	($\phi_{B\&P\_E1}$) B&P solution	$[(\phi_{B\&P\_E1} - \phi_{cplex}) / \phi_{cplex}]$ 100%	($\eta_{B\&P\_E1}$) Number of nodes processed	($\kappa_{B\&P\_E1}$) Number of columns	($t_{B\&P\_E1}$) Total runtime (sec)	(t_{Sub_E1}) Total subproblem time (sec)
1	2	5	21,961	21,961	21,961	0	5	13	0.98	0.63
2	2	10	38,776	38,776	38,776	0	5	37	7.27	6.67
3	4	5	24,416	24,416	24,393	–0.09	291	372	78.36	77.03
4	4	10	48,287	47,809	47,809	–0.99	809	1938	1196.67	1097.09
5	6	5	44,377	44,275	44,275	–0.23	1971	1451	1576.28	1294.16
6	6	10	71,546	70,388	70,388	–1.62	546	1484	>1800	1732
7	8	5	54,558	54,250	54,250	–0.56	1689	1826	>1800	1538.76

Table 10.13. Results using the B&P heuristic to solve the PIDRP with initial columns from tabu search and modified branching rule

Problem no.	τ	n	(ϕ_{cplex}) CPLEX solution	(ϕ_{TS}) Tabu solution	($\phi_{B\&P\_E2}$) B&P solution	$[(\phi_{B\&P\_E2} - \phi_{cplex}) / \phi_{cplex}]$ 100%	($\eta_{B\&P\_E2}$) Number of nodes processed	($\kappa_{B\&P\_E2}$) Number of columns	($t_{B\&P\_E2}$) Total runtime (sec)	(t_{Sub_E2}) Total subproblem time (sec)
1	2	5	21,961	21,961	21,961	0	5	14	2.72	1.05
2	2	10	38,776	38,776	38,776	0	5	37	2.55	2.33
3	4	5	24,416	24,416	24,393	–0.09	239	302	58.36	57.01
4	4	10	48,287	47,809	47,809	–0.99	629	1939	1029.83	949.48
5	6	5	44,377	44,275	44,236	–0.32	429	683	269.44	257.19
6	6	10	71,546	70,388	70,388	–1.62	544	1730	>1800	1747.98
7	8	5	54,558	54,250	54,051	–0.93	929	1694	1005.48	941.14

Table 10.14. Results using the B&P heuristic to solve the PIDRP with initial columns from tabu search, modified branching rule, and initial delivery decisions

Problem no.	τ	n	(ϕ_{cplex}) CPLEX solution	(ϕ_{TS}) Tabu solution	($\phi_{B\&P\_E3}$) B&P solution	[($\phi_{B\&P\_E3} - \phi_{cplex}$) / ϕ_{cplex}] 100%	($\eta_{B\&P\_E3}$) Number of nodes processed	($\kappa_{B\&P\_E3}$) Number of columns	($t_{B\&P\_E3}$) Total runtime (sec)	(t_{Sub_E3}) Total subproblem time (sec)
1	2	5	21,961	21,961	21,961	0	3	13	1.73	1.19
2	2	10	38,776	38,776	38,776	0	3	27	2.72	1.89
3	4	5	24,416	24,416	24,393	–0.09	455	693	141.03	135.09
4	4	10	48,287	47,809	47,809	–0.99	759	1937	1373.80	1198.21
5	6	5	44,377	44,275	44,231	–0.33	345	756	332.58	271.16
6	6	10	71,546	70,388	70,388	–1.62	750	1658	>1800	1657.89
7	8	5	54,558	54,250	54,051	–0.93	667	1345	833.59	748.30

The third experiment examined the performance of the B&P heuristic when model (9-3) was solved in a preprocessing step to determine in which periods deliveries were necessary. The results are summarized in Table 10.14, and when compared to those in Table 10.13, shows no change in solution quality. This was expected; however, we did not see any significant improvement either in terms of the number of nodes processed ($\eta_{B\&P\_E3}$), the number of columns generated ($\kappa_{B\&P\_E3}$), or total runtime ($t_{B\&P\_E3}$). For the seven test problems, model (9-3) was solved in a matter of seconds.

The final option investigated involved the implementation of the rounding heuristic described in Section 9.3. After initial testing, we found that calling this heuristic too frequently provided only marginal improvement between iterations because, when using depth-first search, the upper bounds associated with nearby nodes in the B&B tree change little. Accordingly, we found that a good strategy was to call the heuristic at every 10th node. The corresponding procedure is denoted by E_4 and the results are presented in Table 10.15. When compared to the results in Table 10.13 (the best up to this point), we see that $\phi_{B\&P\_E4}$ is 0.8% less than $\phi_{B\&P\_E2}$ on average. In addition, the number of nodes processed ($\eta_{B\&P\_E4}$) decreased by 20.65%, number of columns generated ($\kappa_{B\&P\_E4}$) decreased by 17.8%, and the total runtime ($t_{B\&P\_E4}$) decreased slightly by 0.7%.

Table 10.15. Results using the B&P heuristic to solve the PIDRP with initial columns from tabu search, modified branching rule, initial delivery decisions, and rounding heuristic (every 10 nodes)

Problem no.	τ	n	(ϕ_{cplex}) CPLEX solution	(ϕ_{TS}) Tabu solution	($\phi_{B\&P\_E4}$) B&P solution	$[(\phi_{B\&P\_E4} - \phi_{cplex}) / \phi_{cplex}]$ 100%	($\eta_{B\&P\_E4}$) Number of nodes processed	($\kappa_{B\&P\_E4}$) Number of columns	($t_{B\&P\_E4}$) Total runtime (sec)	(t_{Sub_E4}) Total subproblem time (sec)
1	2	5	21,961	21,961	21,961	0	3	13	1.89	1.83
2	2	10	38,776	38,776	38,776	0	3	27	3.25	1.72
3	4	5	24,416	24,416	24,393	0	455	693	223.22	158.90
4	4	10	48,287	47,809	47,555	–1.52	611	1937	1305.48	992.37
5	6	5	44,377	44,275	43,288	–2.45	320	731	379.13	298.96
6	6	10	71,546	70,388	70,373	–1.64	485	1098	>1800	828.28
7	8	5	54,558	54,250	52,449	–3.87	329	761	426.39	331.04

After examining the statistics in Tables 10.12 – 10.15, it is evident that initializing $\mathcal{MP}$ with columns from the solution of the tabu search algorithm, implementing the modified branching scheme, and periodically calling the rounding heuristic can lead to a measurable improvement in solution quality and algorithmic efficiency. Although the preprocessing feature didn't provide any noticeable advantage, it could still have some impact on the performance of the B&P algorithm since its effectiveness depends on the actual data. In addition, its computational costs are low, at most 40 seconds for the largest instances. Consequently, all enhanced features discussed in this section will be used in the next section where we try to solve a wide range of problem instances.

10.3.2 Results with standard data sets

In this section, we provide test results for our B&P heuristic, E_4, which includes all four enhancements, for a series of problem instances derived from the three data sets created by Boudia et al. (2007). The original data sets contain 30 instances of 50, 100 and 200 customer problems, all with a 20-period planning horizon and holding costs $h^P = 1$, $h_i^C = 0$ for all $i \in N$. Customers were randomly located on a 100×100 Euclidean grid.

As expected, none of these instances could not be solved optimally, at least within 24 hours, with either CPLEX or our exact B&P algorithm. Therefore, we selectively created five data sets from the original 150 problems to test the limits of our methodology. The first data set was created from the 50 customer problems, the second and third from the 100 customer problems, and the fourth and fifth from the 200 customer problems. Each contains 20 instances.

The first step was to fix the number of customers $n \in \{10, 20, 30, 40, 50\}$ and the number of time periods $\tau \in \{2, 4, 6, 8\}$. By selecting a range of values from these sets, an instance has anywhere from 2 to 8 planning periods and 10 to 50 customers. For each customer i, demand was uniformly distributed between 0 and the storage capacity $I^C_{\max .i}$, and for each period t the probability of that demand being realized varied uniformly in the interval [0.5, 1]. The value of $I^C_{\max .i}$ was a function of i and the number of customers in the specific data set.

Because these new data sets have fewer customers than the original, the number of vehicles was proportionally adjusted downward as follows: $\theta_{10} = 1$, $\theta_{20} = 2$, $\theta_{30} = 3$, $\theta_{40} = 4$, $\theta_{50} = 5$ for corresponding $n \in \{10, 20, 30, 40, 50\}$. Vehicle capacities were $Q_{50} = 8000$, $Q_{100} = 8000$, $Q_{200} = 8000$ for the data sets with 50, 100 and 200 customers, respectively. The maximum storage capacity for customer i, $I^C_{\max .i}$, for data sets 1, 2 and 3 follow the default values from the original data sets. For data sets 4 and 5, $I^C_{\max .i}$ is set to 1000 for all customers.

In the testing, a maximum of 3600 seconds was allowed for a run. In a few cases, termination took place prematurely when the out-of-memory error appeared. The E_4 heuristic was used for all but the largest instances; for data sets with either $n = 40$ and $\tau \geq 6$ or $n \geq 50$, model (9-2) was replaced with model (9-1) to determine delivery quantities prior to calling the VRP subroutine. This modification was necessary because the amount of time required to solve model (9-2) alone at the root node did not permit column generation to be completed within the allotted 3600 seconds. Table 10-16

summarizes the average results for all data sets. The full set of results can be found in Appendix A.

Table 10.16. Summary of results for the B&P heuristic

Data set	$[(\bar{\phi}_{\text{B\&P\_E4}} - \bar{\phi}_{\text{TS}}) / \bar{\phi}_{\text{TS}}]$ 100%	$[(\bar{\phi}_{\text{B\&P\_E4}} - \bar{\phi}_{\text{cplex}}) / \bar{\phi}_{\text{cplex}}]$ 100%	$(\eta_{\text{B\&P\_E4}})$ Average number of nodes	$(\kappa_{\text{B\&P\_E4}})$ Average number of columns	$(t_{\text{B\&P\_E4}})$ Average runtime (sec)	$[(\bar{t}_{\text{B\&P\_E4}} - \bar{t}_{\text{cplex}}) / \bar{t}_{\text{cplex}}]$ 100%
1	–3.94%	–14.72%	40	776	960	–69.84%
2	–3.62%	–12.18%	17	518	703	–74.91%
3	–2.54%	–11.6%	63.2	747	1220	–62.35%
4	–0.59%	–7.33%	3.4	195	188	–93.82%
5	–0.2%	–6.47%	3.3	256	203	–93.45%

In Table 10.16, column 2 identifies the average percentage gap between the solutions from the B&P heuristic ($\bar{\phi}_{\text{B\&P\_E4}}$) and those from tabu search ($\bar{\phi}_{\text{TS}}$). Column 3 gives the gap between $\bar{\phi}_{\text{B\&P\_E4}}$ and the average CPLEX solution ($\bar{\phi}_{\text{cplex}}$). Columns 4 and 5 list the average number of nodes in search tree and the average number of columns generated, respectively. The last column reports the average gap between the B&P heuristic runtime ($\bar{t}_{\text{B\&P\_E4}}$) and the runtime from CPLEX ($\bar{t}_{\text{cplex}}$). In almost all instances, CPLEX consumed the full 3600 seconds while tabu search always terminated in less than 1 minute.

The analysis indicates that, on average, the B&P heuristic improved the tabu search solution by 2.18% and was approximately 10% better than $\bar{\phi}_{\text{cplex}}$. In addition, $\bar{t}_{\text{B\&P\_E4}}$ was 78.87% less than $\bar{t}_{\text{cplex}}$, which demonstrates the relative speed and accuracy of the proposed methodology.

We analyzed the results further by dividing the five data sets into two groups based on storage capacity at the customer sties $I^{C}_{\max .i}$ and the average customer demand. The first group consisted of data sets 1, 2 and 3 where the average of $I^{C}_{\max .i}$ is 800 and the

average demand is 225 per day. The second group consisted of data sets 4 and 5 where the average of $I^C_{\max .i}$ is 1000 and the average demand is 166. While the storage capacity $I^C_{\max .i}$ for group 2 is 25% higher than that of group 1, the average demand for group 2 is 26.22% less than that of group 1. This suggests that the instances in group 2 should be easier to solve than those in group 1 because the vehicle capacity and storage capacity constraints are more relaxed for group 2. The comparative statistics presented in Table 10.17 confirm this observation.

Table 10.17. Summary of results for the B&P heuristic by group

Group	$[(\bar{\phi}_{\text{B\&P\_E4}} - \bar{\phi}_{\text{TS}}) / \bar{\phi}_{\text{TS}}]$ 100%	$[(\bar{\phi}_{\text{B\&P\_E4}} - \bar{\phi}_{\text{cplex}}) / \bar{\phi}_{\text{cplex}}]$ 100%	$(\eta_{\text{B\&P\_E4}})$ Average number of nodes	$(\kappa_{\text{B\&P\_E4}})$ Average number of columns	$[(\bar{t}_{\text{B\&P\_E4}} - \bar{t}_{\text{cplex}}) / \bar{t}_{\text{cplex}}]$ 100%
1	–3.36%	–12.83%	40	680	–69%
2	–0.4%	–7.04%	3.4	226	–93.64%

Looking at the values in columns 2 and 3, we see that both tabu search and CPLEX performed better on the instances in group 2 than on the instances in group 1. In particular, the average gap between $\bar{\phi}_{\text{B\&P\_E4}}$ and $\bar{\phi}_{\text{TS}}$ is 88.27% less and the average gap between $\bar{\phi}_{\text{B\&P\_E4}}$ and the $\bar{\phi}_{\text{cplex}}$ is 45.18% less. The results also show that the B&P heuristic is sensitive to the complexity of the data sets. For the instances in group 2, the average number of nodes and the average number of columns generated are respectively 91.64% and 66.85% less than the corresponding values from group 1.

In summary, the B&P heuristic was shown to provide high quality solutions to PIDRP instances with up to 50 customers and 8 time periods within a reasonable amount of time. For many of these instances, CPLEX failed to find even feasible solutions.

Chapter 11

Discussion and Future Direction

Good solutions to the PIDRP can yield substantial benefits throughout the supply chain as manufacturers and customers become more integrated. A decade ago, the primary challenges facing manufacturers were establishing online communications and delivery channels. Keeping too much inventory was discouraged and warehouses equaled waste. Today, there is a shift in attitude that allows for increased inventory levels where necessary. Lean inventory is a luxury many companies can no longer afford because timely deliveries depend on a far wider range of factors than can be predicted or controlled. If your supply chain is global, those factors include international transportation providers, as well as infrastructure with varying degrees of quality and import/export regulations. If your supply chain is domestic, you're still affected because certain transportation lanes are more crowded than ever, and the competition for transportation services has become even more intense.

In this dissertation, we have provided an efficient reactive tabu search algorithm for finding high quality solutions to the PIDRP as measured by the optimality gap for small instances and by the solution to our lower bounding model otherwise. Although we included a path relinking feature and several theoretical ways of improving the lower bound, none was very effective. We also have provided a decomposition approach to PIDRP. Several enhanced features such as heuristic column generation and rounding heuristic algorithm were successfully implemented. From the computational results the algorithm provides both speed and solution quality for PIDRP.

If there is room for improvement in the methodology, one place to begin is with the two sets of inventory constraints. Adapting cut generation procedures for the capacitated lot-sizing problem offers several opportunities for tightening the LP relaxation of the master problem. This strategy would lead to a branch-and-price-and-cut

algorithm. A second option is to cluster the customers first and then solve the resultant production-inventory-distribution problem and routing problems separately. Finally, taking into account uncertainty represents an ongoing challenge for production planners. In industries where markets shift regularly, capturing the stochastic nature of demand in a manageable model would be highly advantageous for real-time control. Each of these ideas offers computational challenges that the research community has yet to address.

Appendices

APPENDIX A: ADJUSTMENT OF PRODUCTION LEVELS

As mentioned in Section 4.6, it may be possible to improve the tabu search results when evaluating candidate moves by adjusting the production levels. In the implementation, a linear program is solved every 5 iterations to find the optimal values of $\bar{p}_t$. In the remaining cases, we adjust production levels by calling *Production_Level_Adjustment_Algorithm*. The inputs to this algorithm are the production levels $\bar{p}_t$ associated with the most recent tabu search solution and the output generated by *Find_ Adjustment_Period_Algorithm*, which is called when checking the feasibility of a solution.

For a move that involves customer i_1 in period t_1 and customer i_2 in period t_2, $t_1 < t_2$, let $\bar{\eta}_{i_1 t_1}$ be the number of items that were originally scheduled to be delivered to customer i_1 in period t_1 that have been rescheduled for delivery in period t_2, and let $\bar{w}_{i_2 t_2}$ be the number of items to be delivered to customer i_2 in period t_1 after the move. The logic included in the primary algorithms called in the adjustment of production levels is given below. The first step is to check feasibility of a candidate move. Only the inputs and outputs are stated for this algorithm; more detail can be found in Nananukul (2008).

Feasibility_Check_Algorithm, complexity $O(\tau + n)$.

Input: Tabu search solution $\bar{p}_t$ at current iteration, periods t_1 and t_2 ($t_1 < t_2$), customer i_1 and i_2, demand of customer i_1 for all $t \le t_1$, demand of customer i_2 from all $t \le t_2$, delivery amounts $\bar{w}_{i_1 t}$, $t \le t_1$ and $\bar{w}_{i_2 t}$, $t \le t_2$, and swap amounts $\bar{\eta}_{i_1 t_1}$ and $\bar{w}_{i_2 t_2}$.

Output: Output flag: <true> = feasible or <false> = not feasible

If output flag = <true>, then return number of decrease adjustment periods, decrease adjustment periods, decrease adjustment amounts, number of increase adjustment periods, increase adjustment periods, and increase adjustment amounts

\\ comment: The return values are the output from *Find_ Adjustment_Period_Algorithm*.

Find_Adjustment_Period_Algorithm, complexity $O(\tau)$.

Input: Period t^*, search type (1 = search for period(s) $t < t^*$ such that production level needs to be decreased, 2 = search for period(s) $t < t^*$ such that the production level needs to be increased), adjustment quantity AQ , and tabu search solution output ($\overline{p}_t$) at current tabu search iteration

\\ comment: $t^* = t_1$ or t_2 : $AQ = \overline{\eta}_{i_1 t_1} - \overline{w}_{i_2 t_2}$. Only positive value of adjustment quantity AQ is used in the algorithm.

Output: Feasible flag (if feasible period(s) is found, then return "1"; otherwise return "-1")

If search type = 1, then return number of decrease adjustment periods, decrease adjustment periods, and decrease adjustment amounts; otherwise return number of increase adjustment periods, increase adjustment periods, and increase adjustment amounts.

NumAdjPeriods = 1
For $t = t^* - 1, \ldots, 0$
{
  If ($\overline{p}_t > 0$)
  \\ comment: Only period(s) t with $\overline{p}_t > 0$ are considered in the adjustment.
    If (Search type = 1)
      $TAQ = AQ - \overline{p}_t$
    else
      $TAQ = AQ - \left(I^P_{\max} - \overline{p}_t \right)$ \\ TAQ stores remaining adjustment quantity after considering adjustment in period t.
    end if
    If ($TAQ \leq 0$) \\ In this case it means period t is the last period that needs to be considered.

AdjPeriods[NumAdjPeriods] = t
AdjAmount[NumAdjPeriods] = AQ
NumAdjPeriods = NumAdjPeriods + 1
Set feasible flag = 1
If (search type = 1)
Store results in NumDecAdjPeriods, DecAdjPeriods[n];
n = 1,...,NumDecAdjPeriods, DecAdjAmount[n];
n = 1,...,NumDecAdjPeriods
else
Store results in NumIncAdjPeriods, IncAdjPeriods[n];
n = 1,...,NumIncAdjPeriods, IncAdjAmount[n];
n = 1,…,NumIncAdjPeriods
end if
return
else
AdjPeriods[NumAdjPeriods] = t
If (Search type = 1)
AdjAmount[NumAdjPeriods] = $\overline{p}_t$
else
AdjAmount[NumAdjPeriods] = $I^P_{\max} - \overline{p}_t$
end if
NumAdjPeriods = NumAdjPeriods + 1
end if
end if
$AQ = TAQ$
}
If ($TAQ > 0$) \\ In this case it means that the adjustment is not feasible.
Set feasible flag = -1 and return
end if

After the feasibility check is done, the *move_value* for each feasible candidate is calculated by calling *Move_Value_Algorithm*. Because the logic is straightforward, we only give the inputs and outputs.

Move_Value_Algorithm, complexity $O(n^3)$

Input: Periods t_1 and t_2 ($t_1 < t_2$), costs of current VRP solutions in period t_1 and t_2 $\left(C^{VRP}_{t_1}, C^{VRP}_{t_2}\right)$, customer i_1 and i_2, and swap amount $\overline{\eta}_{i_1 t_1}$ (i.e., number of items rescheduled from period t_1 to period t_2 for customer i_1), number of items that were to be delivered to customer i_2 in period t_2 but have been rescheduled for

delivery in period t_1 ($\overline{w}_{i_2 t_2}$), adjustment quantity AQ, number of decrease adjustment periods, decrease adjustment periods, decrease adjustment amounts, number of increase adjustment periods, increase adjustment periods, and increase adjustment amounts

Output: *move_value*

Once all candidate moves are evaluated, the one with the best *move_value* is selected. Finally, the *Production_Level_Adjustment_Algorithm* is called.

Production_Level_Adjustment_Algorithm

Input: Tabu search solution output ($\overline{p}_t$) at current tabu search iteration, Number of decrease adjustment periods, decrease adjustment periods, decrease adjustment amounts, number of increase adjustment periods, increase adjustment periods, and increase adjustment amounts

Output: Updated values of $\overline{p}_t$

Step 1. Update production levels $\overline{p}_t$ in each period t where the new (tabu search) solution indicates a decrease in production level.

For n = 1,…, NumDecAdjPeriods

$$\overline{p}_{\text{DecAdjPeriods}[n]} \leftarrow \overline{p}_{\text{DecAdjPeriods}[n]} - \text{DecAdjAmount}[n]$$

Step 2. Update production levels $\overline{p}_t$ in each period t where the new solution indicates an increase in production level.

For n = 1,…, NumIncAdjPeriods

$$\overline{p}_{\text{IncAdjPeriods}[n]} \leftarrow \overline{p}_{\text{IncAdjPeriods}[n]} - \text{IncAdjAmount}[n]$$

Example (continued). Considering the example in Figures 4.3, assume that the current solution from the allocation model gives $\overline{p}_1 = 66$, $\overline{p}_2 = 0$ and $\overline{p}_3 = 0$. Base on a swap move in Figure 4.3, i_1 = 3, i_2 = 1, t_1 = 2 and t_2 = 3, respectively. Following Step 1 of the above algorithm, NumDecAdjPeriods = 1, DecAdjPeriods[0] = 1, DecAdjAmount[0] =

4, NumIncAdjPeriods = 1, IncAdjPeriods[0] = 1, IncAdjAmount[0] = 4. Continuing, $\overline{p}_1$ is updated to 66 – 4 = 62. In Step 2, $\overline{p}_1$ is updated to 62 + 4 = 66. As a result the swap move does not change the production level in period 1.

For the example in Figures 4.4, assume that the current solution from the allocation model gives $\overline{p}_1 = 64$, $\overline{p}_2 = 2$ and $\overline{p}_3 = 0$. Based on the transfer move in Figure 4.4, $i_1 = 0$, $i_2 = 2$, $t_1 = 2$ and $t_2 = 3$, respectively. Following Step 1 of *Feasibility_Check_Algorithm*, NumDecAdjPeriods = 2, DecAdjPeriods[0] = 2, DecAdjAmount[0] = 2, DecAdjPeriods[1] = 1, DecAdjAmount[1] = 2, NumIncAdjPeriods = 1, IncAdjPeriods[0] = 1, IncAdjAmount[0] = 4. Following Step 1 of *Production_Level_Adjustment_Algorithm*, $\overline{p}_2$ is updated to 2 – 2 = 0. In Step 2, $\overline{p}_1$ is updated to 64 – 2 + 4 = 66.

APPENDIX B: COMPUTATIONAL RESULTS OF HEURISTIC B&P ALGORITHM

Table B1. Results for data set 1 using CPLEX and tabu search to solve the PIDRP

Problem no.	τ	n	(ϕ_{cplex}) CPLEX solution	(t_{cplex}) CPLEX runtime (sec)	(ϕ_{TS}) Tabu search solution	(t_{TS}) Tabu search runtime (sec)
1	2	10	59,192	0.77	59,192	0.46
2	4	10	63,565	6.32	63,565	0.5
3	6	10	81,901	>3600	79,890	9.27
4	8	10	*	>3600	100,959	61
5	2	20	61630	2426.33	62,038	0.43
6	4	20	73,725	>3600	71,500	17.24
7	6	20	100,421	>3600	87,970	10.56
8	8	20	158,643	>3600	115,111	162.2
9	2	30	67,438	>3600	67,703	0.60
10	4	30	86,320	>3600	78,223	30.47
11	6	30	155,314	>3600	100,605	73.50
12	8	30	218,325	>3600	143,272	88.34
13	2	40	88,782	>3600	75,331	0.89
14	4	40	145,502	>3600	101,095	101.32
15	6	40	*	>3600	142,772	116.20
16	8	40	*	>3600	200,311	192
17	2	50	94,082	>3600	83,312	1.94
18	4	50	*	>3600	108,634	48.12
19	6	50	*	>3600	203,047	121.02
20	8	50	*	>3600	256,617	99.05

Table B2. Results for data set 1 using the B&P heuristic E4 to solve the PIDRP

Problem no.	τ	n	($\phi_{B\&P\_E4}$) B&P solution	($t_{B\&P\_E4}$) Total runtime (sec)	[($\phi_{B\&P\_E4} - \phi_{TS}$) / ϕ_{TS}] 100%	[($\phi_{B\&P\_E4} - \phi_{cplex}$) / ϕ_{cplex}] 100%	Number of nodes processed	Number of columns
1	2	10	59,192	2.2	0	0	1	0
2	4	10	63,565	13.08	0	0	1	0
3	6	10	67,923	12.14	–14.98	–17.07	1	42
4	8	10	78,365	511.73	–22.38	N/A	490	753
5	2	20	62,038	4.59	0	0.66	1	0
6	4	20	68,440	9.98	–4.28	–7.17	1	21
7	6	20	87,970	341.19	0	–12.40	31	547
8	8	20	115,111	1628.44	0	–27.44	75	804
9	2	30	67,703	7.02	0	0.39	1	0
10	4	30	76,500	25.67	–2.20	–11.38	1	31
11	6	30	100,605	>3600	0	–35.22	81	2073
12	8	30	138,732	>3600	–3.17	–36.46	3	557
13	2	40	75,331	3.73	0	–15.15	1	0
14	4	40	96,864	700.33	–4.19	–33.43	15	675
15	6	40	141,449[1]	876.03	–0.93	N/A	9	2455
16	8	40	191,227[1]	>3600	–4.53	N/A	43	3842
17	2	50	83,312[1]	7.78	0	–11.45	1	0
18	4	50	108,634[1]	14.88	0	N/A	1	29
19	6	50	179,498[1]	634.88	–11.60	N/A	9	1069
20	8	50	229,722[1]	>3600	–10.48	N/A	29	2612

Table B3. Results for data set 2 using CPLEX and tabu search to solve the PIDRP

Problem no.	τ	n	(ϕ_{cplex}) CPLEX solution	(t_{cplex}) CPLEX runtime (sec)	(ϕ_{TS}) Tabu search solution	(t_{TS}) Tabu search runtime (sec)
1	2	10	80,333	0.64	80,333	0.48
2	4	10	95,815	>3600	95,086	6.24
3	6	10	132,545	>2821[m]	127,183	16.71
4	8	10	*	>3600	170,281	1.87
5	2	20	88,548	>3600	88,542	0.52
6	4	20	142,847	>3600	122,377	10.52
7	6	20	207,587	>3600	178,096	5.25
8	8	20	255,352	>3600	227,518	5.13
9	2	30	97,814	>3600	96,452	0.6
10	4	30	159,946	>3600	131,453	24.19
11	6	30	282,244	>3600	221,178	18.24
12	8	30	*	>3600	278,383	1.98
13	2	40	102,549	>3600	95,491	0.76
14	4	40	*	>3600	142,480	26.91
15	6	40	*	>3600	233,825	20.08
16	8	40	*	>3600	288,903	6.78
17	2	50	138,234	>3600	102,894	0.96
18	4	50	*	>3600	144,871	21.45
19	6	50	*	>3600	239,250	13.94
20	8	50	*	>3600	325,537	8.01

Table B4. Results for data set 2 using the B&P heuristic E4 to solve the PIDRP

Problem no.	τ	n	($\phi_{B\&P\_E4}$) B&P solution	($t_{B\&P\_E4}$) Total runtime (sec)	[($\phi_{B\&P\_E4} - \phi_{TS}$) / ϕ_{TS}] 100%	[($\phi_{B\&P\_E4} - \phi_{cplex}$) / ϕ_{cplex}] 100%	Number of nodes processed	Number of columns
1	2	10	80,333	3.7	0	0	1	0
2	4	10	85,196	83.7	–10.40	–11.08	1	53
3	6	10	127,183	100.89	0	–4.05	37	146
4	8	10	132,265	638.21	–22.33	N/A	105	245
5	2	20	88,542	8.7	0	–0.01	1	0
6	4	20	122,377	54.84	0	–14.33	1	53
7	6	20	178,096	879.55	0	–14.21	3	189
8	8	20	186,581	1017.38	–17.99	–26.93	10	210
9	2	30	96,452	8.17	0	–1.39	1	0
10	4	30	131,453	288.24	0	–17.81	4	235
11	6	30	214,716	>3600	–2.92	–23.93	6	374
12	8	30	268,375	2890	–3.62	N/A	11	198
13	2	40	95,491	8.77	0	–6.88	1	0
14	4	40	142,480	173.55	0	N/A	1	77
15	6	40	222,990[1]	750.38	–4.63	N/A	15	1214
16	8	40	288,903[1]	2934.28	0	N/A	63	3232
17	2	50	102,894[1]	9.94	0	–25.57	1	0
18	4	50	144,871[1]	189.27	0	N/A	1	710
19	6	50	236,148[1]	3100	–1.30	N/A	81	3173
20	8	50	296,062[1]	226.05	–9.05	N/A	1	257

Table B5. Results for data set 3 using CPLEX and tabu search to solve the PIDRP

Problem no.	τ	n	(ϕ_{cplex}) CPLEX solution	(t_{cplex}) CPLEX Run time (sec)	(ϕ_{TS}) Tabu search solution	(t_{TS}) Tabu search runtime (sec)
1	2	10	80,801	14.88	80,801	0.48
2	4	10	103,272	3.56	103,272	3.62
3	6	10	132,294	>3600	125,447	63.21
4	8	10	*	>3600	162,995	2.10
5	2	20	87,481	>3600	87,481	0.52
6	4	20	130,658	>3600	118,552	125.01
7	6	20	223,791	>3600	190,743	8.81
8	8	20	296,712	>3600	248,850	4.95
9	2	30	101,011	>3600	96,758	0.61
10	4	30	175,784	>3600	138,911	14.73
11	6	30	268,255	>3600	208,856	24.30
12	8	30	*	>3600	275,614	2.65
13	2	40	120,283	>3600	99,124	0.76
14	4	40	*	>3600	137,091	42.58
15	6	40	*	>3600	211,665	30.48
16	8	40	*	>3600	295,742	5.6
17	2	50	*	>3600	103,325	0.94
18	4	50	*	>3600	153,964	12.23
19	6	50	*	>3600	242,112	15.50
20	8	50	*	>3600	317,159	2.04

Table B6. Results for data set 3 using the B&P heuristic E4 to solve the PIDRP

Problem no.	τ	n	($\phi_{B\&P\_E4}$) B&P solution	($t_{B\&P\_E4}$) Total runtime (sec)	[($\phi_{B\&P\_E4} - \phi_{TS}$) / ϕ_{TS}] 100%	[($\phi_{B\&P\_E4} - \phi_{cplex}$) / ϕ_{cplex}] 100%	Number of nodes processed	Number of columns
1	2	10	80,801	7.3	0	0	1	0
2	4	10	97,768	69.95	–5.33	–5.33	3	95
3	6	10	115,237	312	–8.14	–12.89	394	565
4	8	10	126,895	340.69	–22.15	N/A	41	211
5	2	20	87,481	35.83	0	0	1	0
6	4	20	118,552	111.86	0	–9.27	3	131
7	6	20	190,743	480.73	0	–14.77	10	180
8	8	20	238,524	>3600	–4.15	–19.61	1	56
9	2	30	96,758	10.89	0	–4.21	1	0
10	4	30	138,911	325.02	0	–20.98	23	76
11	6	30	206,601	>3600	–1.08	–22.98	14	429
12	8	30	275,614	1002	0	N/A	1	220
13	2	40	99,124	17.55	0	–17.59	1	0
14	4	40	137,091	2694.19	0	N/A	17	1055
15	6	40	202,601[1]	1263	–4.28	N/A	538	1904
16	8	40	285,874[1]	3248.31	–3.34	N/A	45	3492
17	2	50	103,325[1]	17.47	0	N/A	1	0
18	4	50	153,964[1]	64.72	0	N/A	1	180
19	6	50	236,592[1]	>3600	–2.28	N/A	150	3253
20	8	50	317,159[1]	>3600	0	N/A	18	3093

Table B7. Results for data set 4 using CPLEX and tabu search to solve the PIDRP

Problem no.	τ	n	(ϕ_{cplex}) CPLEX solution	(t_{cplex}) CPLEX runtime (sec)	(ϕ_{TS}) Tabu search solution	(t_{TS}) Tabu search runtime (sec)
1	2	10	129,320	1.3	129,320	2.7
2	4	10	130,868	3.38	130,868	1.26
3	6	10	139,281	>3600	139,603	3.92
4	8	10	*	>3600	148,754	4.09
5	2	20	133,550	115.12	133,739	1.29
6	4	20	139,090	204,13	139,090	1.46
7	6	20	148,274	>3600	148,226	0.55
8	8	20	240,499	>3600	185,773	7.71
9	2	30	142,496	>3600	140,145	1.51
10	4	30	149,100	>3600	147,360	12.82
11	6	30	206,208	>3600	177,512	7.06
12	8	30	274,094	>3600	213,431	12.18
13	2	40	175,317	>3600	143,874	1.86
14	4	40	176,678	>3600	159,158	11
15	6	40	*	>3600	195,485	29
16	8	40	*	>3600	236,561	24.44
17	2	50	*	>3600	147,680	2.46
18	4	50	*	>3600	172,942	15.35
19	6	50	*	>3600	207,883	77.6
20	8	50	*	>3600	247,572	26.71

Table B8. Results for data set 4 using the B&P heuristic E4 to solve the PIDRP

Problem no.	τ	n	($\phi_{B\&P\_E4}$) B&P solution	($t_{B\&P\_E4}$) Total runtime (sec)	[($\phi_{B\&P\_E4} - \phi_{TS}$) / ϕ_{TS}] 100%	[($\phi_{B\&P\_E4} - \phi_{cplex}$) / ϕ_{cplex}] 100%	Number of nodes processed	Number of columns
1	2	10	129,320	7.82	0	0	1	0
2	4	10	130,868	12.28	0	0	1	6
3	6	10	139,603	65.56	0	0.23	11	50
4	8	10	145,346	83.22	–2.29	N/A	23	290
5	2	20	133,739	1.14	0	0.14	1	0
6	4	20	139,090	2.53	0	0.00	1	6
7	6	20	148,226	13.19	0	–0.03	1	22
8	8	20	185,773	1737.42	0	–22.76	17	1059
9	2	30	140,145	6.31	0	–1.65	1	0
10	4	30	147,360	5.14	0	–1.17	1	6
11	6	30	177,512	113.38	0	–13.92	1	152
12	8	30	196,661	1141.30	–7.86	–28.25	1	478
13	2	40	143,874	2.98	0	–17.93	1	0
14	4	40	159,158	6.33	0	–9.92	1	6
15	6	40	195,485[1]	29.89	0	N/A	1	176
16	8	40	236,561[1]	29.53	0	N/A	1	174
17	2	50	147,680[1]	0.84	0	N/A	1	0
18	4	50	169,963[1]	19.36	–1.72	N/A	1	61
19	6	50	207,883[1]	25.03	0	N/A	1	82
20	8	50	247,572[1]	453	0	N/A	1	1334

Table B9. Results for data set 5 using CPLEX and tabu search to solve the PIDRP

Problem no.	τ	n	(ϕ_{cplex}) CPLEX solution	(t_{cplex}) CPLEX runtime (sec)	(ϕ_{TS}) Tabu search solution	(t_{TS}) Tabu search runtime (sec)
1	2	10	128,448	0.9	128,448	2.57
2	4	10	132,235	1.97	132,235	0.83
3	6	10	138,990	>3600	138,990	3.06
4	8	10	*	>3600	140,941	4.52
5	2	20	133,674	908.39	133,674	0.93
6	4	20	142,006	>3600	141,283	10.90
7	6	20	171,683	>3600	156,564	4.39
8	8	20	201,447	>3600	173,754	12.88
9	2	30	140,069	>3600	139,410	1.19
10	4	30	166,819	>3600	150,660	2.17
11	6	30	216,694	>3600	181,674	64.39
12	8	30	275,926	>3600	230,462	17.14
13	2	40	153,040	>3600	144,641	1.60
14	4	40	179,449	>3600	163,015	11.85
15	6	40	*	>3600	195,961	30.16
16	8	40	*	>3600	231,962	24.83
17	2	50	*	>3600	148,843	1.81
18	4	50	*	>3600	169,834	48.47
19	6	50	*	>3600	207,824	27.18
20	8	50	*	>3600	257,578	28.72

Table B10. Results for data set 5 using the B&P heuristic E4 to solve the PIDRP

Problem no.	τ	n	($\phi_{B\&P\_H4}$) B&P solution	($t_{B\&P\_H4}$) Total runtime (sec)	[($\phi_{B\&P\_H4} - \phi_{TS}$) / ϕ_{TS}] 100%	[($\phi_{B\&P\_H4} - \phi_{cplex}$) / ϕ_{cplex}] 100%	Number of nodes processed	Number of columns
1	2	10	128,448	8.02	0	0	1	0
2	4	10	132,235	12.19	0	0	1	6
3	6	10	138,990	20.16	0	0	11	29
4	8	10	140,941	19.06	0	N/A	25	17
5	2	20	133,674	36.21	0	0	1	0
6	4	20	141,283	1.94	0	–0.51	1	6
7	6	20	156,564	22.55	0	–8.81	1	55
8	8	20	173,754	55.12	0	–13.75	1	94
9	2	30	139,410	1.17	0	–0.47	1	0
10	4	30	150,660	5.06	0	–9.69	1	6
11	6	30	181,674	49.55	0	–16.16	1	55
12	8	30	221,066	3214.27	–4.08	–19.88	3	567
13	2	40	144,641	3.88	0	–5.49	1	0
14	4	40	163,015	58.31	0	–9.16	1	49
15	6	40	195,961[1]	116.91	0	N/A	3	494
16	8	40	231,962[1]	127.46	0	N/A	7	2813
17	2	50	148,843[1]	3.13	0	N/A	1	0
18	4	50	169,834[1]	12.20	0	N/A	1	15
19	6	50	207,824[1]	32.80	0	N/A	1	116
20	8	50	257,578[1]	271.30	0	N/A	3	789

* Cplex cannot find integer solution

[1] means the subproblem model 1 is used

[m] out of memory

Bibliography

Abdelmaguid, T.F. and Dessouky M.M. 2006, A genetic algorithm approach to the integrated inventory-distribution problem. *International Journal of Production Research* 44(21) 4445-4464.

Assad, A., Golden B., Dahl R. and Dror M. 1983. Evaluating the effectiveness of an integrated system for fuel delivery. in Eatman, J. (editor). *Proceedings of the Southeast TIMS Nineteenth Annual Meeting*. Myrtle Beach. SC. 153-160.

Anily, S. and Federgruen A. 1990. One warehouse multiple retailer system with vehicle routing costs. *Management Science* 36(1) 92-114.

Anily, S. and Federgruen A. 1993. Two-echelon distribution systems with vehicle routing costs and central inventories. *Operations Research* 41(1) 37-47.

Bard, J.F., Huang L., Jaillet P. and Dror M. 1998. A decomposition approach to the inventory routing problem with satellite facilities. *Transportation Science* 32(2) 189-203.

Bard, J.F., Kontoravdis G. and Yu G. 2002. A branch-and-bound procedure for the vehicle routing problem with time windows. *Transportation Science* 36(2) 250-269.

Bard, J.F and Rojanasoonthon S. 2006. A branch & price algorithm for parallel machine scheduling with time windows and job priorities. *Naval Research Logistics* 53(1) 24-44.

Baudin, M. 2004. Lean Logistics: The Nuts and Bolts of Delivering Materials and Goods. Productivity Press. New York.

Bertazzi, L., Paletta G. and Speranza M.G. 2002. Deterministic order-up-to level policies in an inventory routing problem. *Transportation Science* 36(1) 119-132.

Boudia, M., Louly M.A.O. and Prins C. 2006. A memetic algorithm with population management for a production-distribution problem. in Doglui, A., Morel G. and Pereira C.E. (editors). *12th IFAC Symposium on Information Control Problems in Manufacturing*. May 17-19. Saint-Etienne. France. 3 541-546.

Boudia, M., Louly M.A.O. and Prins C. 2007. A reactive grasp and path relinking for a combined production-distribution problem. *Computers & Operations Research* 34(11) 3402-3419.

Cachon, G.P. 1999. Managing supply chain demand variability with scheduled ordering

policies. *Management Science* 45 843–856.

Carlton, B.W. and Barnes J.W. 1996. Solving the traveling salesman problem with time windows using tabu search. *IIE Transactions on Scheduling & Logistics* 28(8) 617-629.

Cetinkaya, S., Mutlu F. and Lee C.-Y 2006. A comparison of outbound dispatch policies for integrated inventory and transportation decisions. *European Journal of Operational Research* 171(3) 1094-1112.

Chan, L.M.A., Federgruen A. and Simchi-Levi D. 1998. Probabilistic analyses and practical algorithms for inventory-routing models. *Operations Research* 46(1) 96-106.

Chien, T.W., Balakrishnan A. and Wong R.T. 1989. An integrated inventory allocation and vehicle routing problem. *Transportation Science* 23(2) 67-76.

Chand, S., Hsu V.N., Sethi S. and Deshpande V. 2007. A dynamic lot sizing problem with multiple customers: customer-specific shipping and backlogging costs. *IIE Transactions on Scheduling & Logistics* 39(11) 1059-1069.

Chandra, P. and Fisher M.L. 1994. Coordination of production and distribution planning. *European Journal of Operational Research* 72(3) 503-517.

Cheung, K.L. and Zhang S.H. 2008. Balanced and synchronized ordering in supply chains. *IIE Transactions on Scheduling & Logistics* 40(1) 1-11.

Choi, E. and Tcha D.-W. 2007. A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research* 34(7) 2080-2095.

Clark, A.J. and Scarf H. 1960. Optimal policies for a multiechelon inventory problem. *Management Science* 6(4) 475-490.

Cordeau, J.-F., Gendreau M. and Laporte G. 1997. A tabu search heuristic for periodic and multi-depot vehicle routing problem. *Networks* 30(2) 105-119.

Dantzig, G.B. and Ramser J.H. 1959. The truck dispatching problem. *Management Science* 6(1) 80-91.

Desrochers, M., Desrosiers J., Solomon M.M. 1992. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research* 40(22) 342–354.

Dror, M. and Ball M. 1987. Inventory/routing: reduction from an annual to a short period problem. *Naval Research Logistics Quarterly* 34(4) 891-905.

Dror, M., Ball M. and Golden B. 1985. A computational comparison of algorithms for the inventory routing problem. *Annals of Operations Research* 4(1-4) 3-23.

Evans, J.R. 1985. An efficient implementation of the Wagner-Whitin algorithm for dynamic lot-sizing. *Journal of Operations Management* 5(2) 229-235.

Federgruen, A. and Tzur M. 1991. A simple forward algorithm to solve general dynamic lot sizing models with n periods in $O(n\log n)$ or $O(n)$ time. *Management Science* 37(8) 909-925.

Federgruen, A. and Tzur M. 1999. Time-partitioning heuristics: application to one warehouse, multiitem, multiretailer lot-sizing problems. *Naval Research Logistics* 46(5) 463-486.

Federgruen, A. and Zipkin P. 1984. A combined vehicle routing and inventory allocation problem. *Operations Research* 32(5) 1019-1037.

Fukasawa, R., Longo H., Lysgaard J., Poggi de Aragão M., Reis M., Uchoa E. and Werneck R.F. 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming* 106(3) 491-511.

Fumero, F. and Vercellis C. 1999. Synchronized development of production, inventory, and distribution schedules. *Transportation Science* 33(3) 330-340.

Gaudioso, M. and Paletta G. 1992. A heuristic for the periodic vehicle routing problem. *Transportation Science* 26(2) 86-92.

Gaur, V. and Fisher M.L. 2004. A periodic inventory routing problem at a supermarket chain. *Operations Research* 52(6) 813-822.

Glover, F. 1989. Tabu search – part I. *ORSA Journal on Computing* 1(3) 190-206.

Glover, F. 1990. Tabu search – part II. *ORSA Journal on Computing* 2(1) 4-32.

Glover, F. and Laguna M. 1997. *Tabu search.* Kluwer. Norwell. MA.

Glover, F., Jones G., Karney D., Klingman D. and Mote J. 1979. An integrated production, distribution, and inventory planning system. *Interfaces* 9(5) 21-35.

Golden, B., Assad A. and Dahl R. 1984. Analysis of a large scale vehicle routing problem with an inventory component. *Large Scale Systems* 7(2-3) 181-190.

Gendreau, M., Hertz A. and Laporte G. 1992. New insertion and post-optimization procedures for the traveling salesman problem. *Operations Research* 40(6) 1086-1094.

Gendreau, M., Hertz A. and Laporte G. 1994. A tabu search heuristic for the vehicle routing problem. *Management Science* 40(10) 1276-1290.

Gutiérrez, J., Sedeño-Noda A., Colebrook M. and Sicilia J. 2002. A new characterization for the dynamic lot size problem with bounded inventory. *Computers & Operations Research* 30(3) 383-395.

Gutiérrez, J., Sedeño-Noda A., Colebrook M. and Sicilia J. 2007. A polynomial algorithm for the production/ordering planning problem with limited storage. *Computers & Operations Research* 34(4) 934-937.

Herer, Y.T. and Roundy R. 1997. Heuristics for a one-warehouse multiretailer distribution problem with performance bounds. *Operations Research* 45(1) 102-115.

Herer, Y.T., Tzur M. and Yucesan E. 2006. The multilocation transshipment problem. *IIE Transactions on Scheduling & Logistics* 38(3) 185-200.

Irnich, S., Funke B. and Grunert T. 2006. Sequential search and its application to vehicle-routing problems. *Computers & Operations Research* 33(8) 2405-2429.

Jayaraman, V. and Pirkul H. 2001. Planning and coordination of production and distribution facilities for multiple commodities. *European Journal of Operational Research* 133(2) 394-408.

Kontoravdis, G. and Bard J.F. 1995. A GRASP for the vehicle routing problem with time windows. *ORSA Journal on Computing 7*(1) 10-23.

Laporte, G. and Nobert Y. 1987. Exact algorithms for the vehicle routing problem. *Survey in Combinatorial Optimizatio.* North-Holland. Amsterdam. 147-184.

Laporte, G. 1992. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research* 59(3) 345-358.

Lei, L., Liu S., Ruszczynski A. and Park S. 2006. On the integrated production, inventory, and distribution routing problem. *IIE Transactions on Scheduling & Logistics* 38(11) 955-970.

Love, S.F. 1973. Bounded production and inventory models with piecewise concave costs. *Management Science* 20(3) 313-318.

Lübbecke, M.E. and Desrosiers, J. 2005. Selected topics in column generation. *Operations Research* 53(6) 1007–1023.

Mourgaya, M. and Vanderbeck F. 2006. Periodic vehicle routing problem: classification and heuristic. *RAIRO Operations Research* 40 169-194.

Mourgaya, M. and Vanderbeck F. 2007. Column generation based heuristic for tactical planning in multi-period vehicle routing. to appear in *European Journal of Operational Research*.

Nemhaurser, G. and Wolsey L. 1988. Integer and Combinatorial Optimization. John Wiley & Sons. New York.

O'Brien, C. and Tang O. (editors) 2006. Integrated enterprise and supply chain management. *International Journal of Production Economics* 101 (special issue).

Ozdamar, L. and Yazgac T. 1999. A hierarchical planning approach for a production-distribution system. *International Journal of Production Research* 37(16) 3759-3772.

Parthanadee, P. and Logendran R. 2006. Periodic product distribution from multi-depots under limited supplies. *Institute of Industrial Engineers Transactions* 38(11) 1009-1026.

Pochet, Y. 1988. Valid inequalities and separation for capacitated economic lot sizing. *Operations Research Letters* 7(3) 109-115.

Pochet, Y. and Wolsey L.A. 1991. Solving multi-item lot-sizing problems using strong cutting planes. *Management Science* 37(1) 53-67.

Pochet, Y. and Wolsey L.A. 1993. Lot-sizing with constant batches: formulation and valid inequalities. *Mathematics of Operations Research* 18(4) 767-785.

Poggi de Aragão, M. and Uchoa E. 2003. Integer program reformulation for robust branch-and-cut-and-price. *Annals of Mathematical Programming in Rio*. Proceedings of Conference Held at Federal University of Rio de Janeiro. COPPE/Sistemas. Búzios. Brazil. 56-61.

Prais, M. and Ribeiro C. 2000. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing* 12(3) 164-176.

Resende, M.G.C. and Ribeiro C.C. 2005. GRASP with path-relinking: Recent advances and applications. in Ibaraki, T., Nonobe K. and Yagiura M. (editors). *Metaheuristics: Progress as Real Problem Solvers*. Springer. New York. 29-63.

Roundy, R. 1985. 98%-effective integer-ratio lot-sizing for one-warehouse multi-retailers systems. *Management Science* 31(11) 1416-1430.

Ryan, D.M. and Foster B.A. 1981. An integer programming approach to scheduling. in A. Wren (editor). *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*. North Holland. Amsterdam. 269-280.

Savelsbergh, M. 1997. A branch-and-price algorithm for the generalized assignment problem. *Operations Research* 45(6) 831-841.

Solomon, M. 1987. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research* 35(2) 254–265.

Van Buer, M.G., Woodruff D.L. and Olson R.T. 1999. Solving the medium newspaper production/distribution problem. *European Journal of Operational Research* 115(2) 237-253.

Vanderbeck, F. 2000. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research* 48(1) 111-128.

Villegas, F.A. and Smith N.R. 2006. Supply chain dynamics: analysis of inventory vs. order öscillations trade-off. *International Journal of Production Research* 44(6) 1037-054.

Wagner, H.M. and Whitin T.M. 1959. Dynamic version of the economic lot size model. *Management Science* 5(1) 89-96.

Wolsey, L.A. 1998. Integer Programming. Willey-Interscience. New York.

Zhao, Q.-H., Wang S.-Y. and Lai K.K. 2007. A partition approach to the inventory/routing problem. *European Journal of Operational Research* 177(2) 786-802.

Vita

Narameth Nananukul was born in Bangkok, Thailand on May 3, 1970, the son of Kasem Nananukul and Chuanchuen Nananukul. After finish his High School diploma in Bangkok, Thailand, in 1988, he entered Chulalongkorn University in Bangkok, Thailand. He received the degrees of Bachelor of Engineering and Master of Engineering in Electrical Engineering from Chulalongkorn University in 1991 and 1994, respectively. In 1994 he was employed as an automation engineer at Seagate Technology. In August 1996, he entered the Graduate School of Texas A&M University and got his Master Degree of Engineering in May 1999. He was employed as an Operations Research analyst at Saitech-Inc in Hazlet, NJ from January 2000 to October 2003.

In August 2004, he started his doctoral study in Operations Research and Industrial Engineering at the University of Texas, Austin.

Permanent address: 9/32 Riverline Place 1, Piboonsongkram Rd., Nonthanburi, Thailand, 11000

This dissertation was typed by Narameth Nananukul.